Revealing Spam Detection: A Hybrid Machine and Deep Learning Approach to Uncovering Key Textual Features

by Mokhtar Ismeil

M.S. in System Engineering, 2015, The George Washington University
M.S. in Telecom Technologies, 2009, Sikkim Manipal University
B.S. in Information Technology & Computing, 2007, The Arab Open University

A Praxis submitted to

The Faculty of
The School of Engineering and Applied Science
of The George Washington University
in partial fulfillment of the requirements
for the degree of Doctor of Engineering

May 18, 2025

Praxis directed by

Amir Etemadi
Associate Professor of Engineering and Applied Science, Committee Member

Joseph Blackford
Associate Professor of Engineering and Applied Science, Committee Member

The School of Engineering and Applied Science of The George Washington University certifies that Mokhtar Ismeil has passed the Final Examination for the degree of Doctor of Engineering as of March 22, 2025. This is the final and approved form of the Praxis

Revealing Spam Detection: A Hybrid Machine and Deep Learning Approach to Uncovering Key Textual Features

Mokhtar Ismeil

Praxis Research Committee:

Amir Etemadi, Associate Professor of Engineering and Applied Science, Praxis Director

Joseph Blackford, Associate Professor of Engineering and Applied Science, Praxis Director

Evan Wehr, Professorial Professor of Engineering and Applied Science, Committee Member

© Copyright 2025 by Mokhtar Ismeil
All rights reserved

Dedication

This Praxis is dedicated to all the people who have supported me throughout this journey. I dedicate this work to my esteemed professors and advisors for their unwavering support and guidance during my research.

To my beloved wife, Hiba, for her endless patience and encouragement. To my father and mother, for their unconditional love and support. And to my son, Jad, who inspires me every day.

Acknowledgements

I would like to express my deepest gratitude to my advisors for their invaluable guidance and support throughout this research.

I would also like to acknowledge Dr. Amir Etemadi, Associate Professor of Engineering and Applied Science, and Dr. Joseph Blackford, Associate Professor of Engineering and Applied Science, as committee members, for their thoughtful contributions and insights.

Additionally, I am grateful to the Doctoral Administrative Team at George Washington University for their support and resources that made this work possible.

Abstract of Praxis

Revealing Spam Detection: A Hybrid Machine and Deep Learning Approach to Uncovering Key Textual Features

Bridging the two problems of achieving a high level of effectiveness in the classification of spam emails and scrutinizing the text of the email that influences the decision, and the classification is the aim of this study. A hybrid machine learning and deep learning model is built to build word matrices of count emails based on punctuation, capitalization, and marked words. Such representations allow the construction of robust automated spam filter systems, wherein system countermeasures can be designed progressively against more advanced spammer intelligence while explaining why certain emails deemed spam were marked as such.

The model transforms emails into structured features, while preserving the fundamental structure differentiating spam from non-spam emails. Results with the Spambase dataset indicate that all evaluated models, including Random Forest, Gradient Boosting, MLP, and a Keras-based neural network, achieved high accuracy and F1 scores, confirming spam classification validity. Further analysis also illustrates how recalibrating the estimates from the classification refined the spam probability during the post-processing stage, and feature resilience analyses showed only slight reductions in performance when excising dominant features. All these results demonstrate the enduring

effectiveness of this model, which is usable in combative email security systems, and illustrate the challenges posed by emerging spam emails.

The framework is able to absorb the impact of gaps in trust and respond to excessive dependence on strict boundaries considering the dynamic nature of email inflow to minimize the adverse effects of spam emails.

Table of Contents

Dedication iv

Acknowledgements v

Abstract of Praxis vi

Table of Contents viii

List of Figures xii

List of Symbols xiii

List of Acronyms xiv

Glossary of Terms xvi

Chapter 1 - Introduction 1

1.1 Background 1

1.2 Spam Issue History 1

1.3 The Evolution from Rule-Based detection Systems to Machine Learning Systems 3

1.4 The Advancement of Deep Learning Technology and Its Impact on Spam Filtering 4

1.5 The Rationale for An Integrated Approach 5

1.6 Issues Relating to Trust and Security within Email Accounts. 6

1.7 Problem Statement 7

1.8 Thesis Statement 9

1.9 Research Objectives 11

1.10 Research Questions and Hypotheses 13

1.10.1 Research Questions: 13

1.10.2 Research Hypotheses: 14

1.11 Implications and Significance 15

1.11.1 Contributions to Knowledge 15

1.11.2 Practical Implications 16

1.11.3 Broader Impact and Future Directions 17

Chapter 2: Literature Review 20

2.1 An Outline of Different Strategies Employed for Spam Detection 20

2.2 Machine Learning Approaches to Spam Detection 23

2.3 Approaches to Spam Detection through Deep Learning 30

2.4 Interpretability and Explainability in Spam Detection 35

2.5 Research Gaps and Future Directions 38

Chapter 3: Methodology 41

3.1 Introduction to the Methodological Framework 41

3.2 Data Preparation 44

3.3 Model Selection and Justification 50

3.3.1 Traditional Machine Learning Models 50

3.3.2 Keras-Based Deep Learning 55

3.4 Hybrid Considerations 58

3.5 Model Training and Hyperparameter Tuning 60

3.5.1 Tuning Parameters for Traditional ML 60

3.5.2 Keras Deep Learning Frameworks data processing 64
3.6 Calibration and Explainability 67
3.7 Cross-Validation and Evaluation Metrics 68
3.8 Probability Calibration 71
3.9 Feature Resilience Testing 75
3.10 Implementation Details 77
3.11 Hardware and Computational Environment 78
Chapter 4: Results 82
4.1 Introduction to the Results Chapter 82
4.2 Initial Exploratory Findings 82
4.3 Baseline Performance of Each Model 88
4.3.1 Models' Classification Report 88
4.3.2 Analysis of Random Forest 89
4.3.3 Examination of Gradient Boosting 90
4.3.4 Analysis of MLP (Classical Neural Network) 91
4.3.5 Analysis of the Deep Learning (Keras-Based Model) 92
4.4 Consolidated Comparison and Observations 93
4.5 Analysis of The Confusion Matrix for All Four Models 94
4.6 ROC Curves Comparison for All Models 96
4.7 Precision-Recall Curves Comparison for All Models 99
4.8 Deep Learning Model - Loss and Accuracy Learning Curves 101
4.9 Outcomes for Calibration of Probabilities 103
4.9.1 Calibration Observations for Ensemble Methods and MLP 103

4.9.2 Deep Learning Model Temperature Scaling 106
4.10 Feature Resilience Results 109

Chapter 5: Discussion and Conclusion 115

5.1 Overview of Key Findings 115

5.2 Addressing Research Questions and Hypotheses 117

5.3 Broader Implications and Benefits 119

5.4 Limitations and Future Directions 121

References 123

Appendix A 132

A: Extended Data and Feature Summaries 132

A.1 Spambase Data Dictionary 132
A.2 Numeric Features Data Snippet 133

Appendix B 134

B. Supplementary Model Performance Visuals 134

B.1 K-Fold CV F1 Scores by Model 134
B.2 Radar Chart Comparing Model Performance 134
B.3 Feature Resilience Test Results (Removing capital_run_length_total) 135
B.4 Summary of Model Metrics (Original vs. After Feature Removal) 136
B.5 Comprehensive Model Performance Table 136
B.6 Calibration Bins (Before vs. After Calibration) 137

List of Figures

Figure 1: Workflow for Predictions and SHAP Explanations..... 42

Figure 2: Overall Spam vs. non-spam Distribution 44

Figure 3: Data Columns extracted from the Spambase Dataset 46

Figure 4: Correlation heatmap for sample features..... 48

Figure 5: SHAP Summary Plot for Gradient Boosting..... 53

Figure 6: Neural Network Architecture for Spam Classification..... 66

Figure 7: Overall Methodology Pipeline 81

Figure 8: Correlation Heatmap for Top 15 Features 83

Figure 9: SHAP Bar Plot for Feature Importance..... 86

Figure 10 : Models' Classification Report 88

Figure 11 : Comparison of Model metrics 93

Figure 12: Confusion matrices of the four models 94

Figure 13: All Models' Curves Comparison 97

Figure 14: Comparison of All Models' Precision-Recall Curves 99

Figure 15: Deep Learning Model - Loss and Accuracy Learning Curves 101

Figure 16: Calibration Curves for Random Forest, Gradient Boosting, and MLP 104

Figure 17: Calibration Curves for Deep Learning Model..... 106

Figure 18: The combined calibration values..... 108

Figure 19: Resilience Test Results for Top 10 Features.....111

Figure 20: Resilience Test Results for 3 Random Features 112

List of Symbols

~	Approximately
Δ	Difference in value
Df	Data Frame object in pandas
e.g.	For example
Exp	Exponential function
/	Fraction
<	Smaller than
>	Greater than
Log	Logarithm function
Max	Maximum value
N_d	Dominant class
N_s	Smaller class
%	Percent
σSM	Softmax function (σ)
∑	Summation
T	Temperature Parameter

List of Acronyms

Acronyms	Definition
AI	Artificial Intelligence
API	Application Programming Interface
ARPANET	Advanced Research Projects Agency Network
AUC	Area Under the Curve
BERT	Bidirectional Encoder Representations from Transformers
Char	Character
CNNs	Convolutional Neural Networks
CPU	Central Processing Unit
EDA	Exploratory Data Analysis
Freq	Frequency
FNNs	Feedforward Neural Networks
GPU	Graphics Processing Unit
K-Fold	K-Fold Cross Validation
LSTM	Long Short-Term Memory

ML	Machine Learning
NaN	Not a Number
ReLU	Rectified Linear Unit
ROC	Receiver Operating Characteristic
RNNs	Recurrent Neural Networks
RMSProp	Root Mean Square Propagation
SHAP	SHapley Additive exPlanations
TF-IDF	Term Frequency-Inverse Document Frequency
XAI	Explainable Artificial Intelligence

Glossary of Terms

AdaGrad: An adaptive gradient method that increases learning performance by setting separate instructional rates for each characteristic. As a result, the need for sparse data and hyperparameter configuration is minimized (Asmelash et al., 2015).

Bag of Words (BoW): In computer vision, it can be used to classify images, videos, robot localization, and more. It is one of the most common methods for categorizing texts and objects (Ameen et al., 2019).

Binary Cross Entropy: This function can be defined in a machine learning context and utilized in binary classification problems. It would be very convenient when each problem could be simplified to a yes or no answer; binary cross entropy works well for situations involving many classification problems simultaneously (Ruby et al., 2020).

Deep learning: It is a subset of machine learning that uses a set of algorithms that processes data in a hierarchical structure made up of many nonlinear interactions (Bhattacharyya et al., 2020).

K-fold Cross-Validation: This method organizes the dataset into K sets of equal size. One set is reserved for training, while the classifier is tested on each set in turn (Langford, 2005).

Naive Bayes: This is a supervised learning model dating back to the early eighteenth century that addresses both binary and multi-class classification tasks (Nokeri, 2021).

SVM: Support Vector Machines (SVMs) function as algorithms for classification and also serve many other tasks such as regression, density estimation, novelty detection, etc. For

the simplest type of two-class classification, SVMs create a hyperplane that gives the maximum possible distance between the two data classes (Xinhua, 2017).

Chapter 1 - Introduction

1.1 Background

Email remains one of the primary means of communication in business, academic, or personal contexts. Nonetheless, the masterful employ of email has come with an increase in unsolicited, fraudulent, and even harmful spam emails. This influx is not only irritating but also extremely dangerous as it conceals phishing, malware, and other forms of cybercrime. Spam is considered a massive threat to cybersecurity; therefore, much worry and creativity must be put into filtering spam, and enhance the current detection mechanisms.

1.2 Spam Issue History

Spam became a serious problem with the birth of ARPANET in the 1970s, but it was not until the 1990s and early 2000s that it began to overrun email accounts in the beginning stages of spam emails. The content was trivial, about basic advertising or mass forwarded emails containing repetitive and dull text. Around this time, automated email sorting systems that depended on keyword blocklisting and basic syntactic rules came into existence. For example, filters were invented to block emails containing the term like "Viagra" or “Money” along with emails from recognized spam domains.

The dramatic rise in spammers comes as no surprise that email security systems had to evolve – and indeed, they did. As spammers started using techniques to sift through email contents, graph manipulation, tactical deception, and outrageous misspellings became

widespread methods for evading filters. Leaving aside the countless reports of growing spam, things started getting out of control in the mid-2000s. It is estimated that, at the peak of the spam war, over 90% of emails were spam, making the detection systems obsolete (Yu, 2011).

The sharp increase in spam emails caused a shift in social paradigms, and the blame can be placed on spam. Spam attacks everybody regardless of age and class; even institutions are victims since they are losing out on internet bandwidth while shelling out money to mitigate the cyber security threat. Many cybercriminals start their careers with impersonation fraud, and as they learn more skills, the severity of their crimes progresses to corporate espionage along with ransomware attacks. Analyzing the negative consequences of spam on the economy is quite disturbing, as there is no cap on the money spent to fight spam. The damages inflicted through file breaches due to spam are endless.

Discerning these two issues serves as a barrier to eliminating spam. First, the thin line that separates spam from legit emails (referred to as 'ham') has narrowed over the years.

Several legitimate emails can pass off as spam because of how they are written or their promotional nature. Hence, it is impossible to set accurate rules. On the other hand, the very presence of spam is a problem in itself, as new methods of filtering evasion are perpetually being invented by spammers. Many researchers have tried to solve this problem; for example, Hemalatha and others argue that spam detection problems must be dealt with using adaptive and opaque detection systems (Hemalatha et al., 2022).

1.3 The Evolution from Rule-Based detection Systems to Machine Learning Systems

The transition from using spam techniques and rule-based systems towards adopting machine learning approaches advanced with adopting procedures such as Naive Bayes classifiers, decision trees, and support vector machines (SVMs). During the early days of machine learning algorithms, these classification algorithms were trained using large datasets of labeled emails. The algorithms embedded into the system attempted to learn the classification of spam and non-spam (ham) emails by studying and measuring different statistical word patterns and frequencies, syntactic structures, and other relevant features.

Those techniques mostly changed on their own and taught themselves. In other words, there was no need for stringent, rigid procedures as the algorithms could be free from boundless data constraints. However, certain gaps were left unfulfilled. The traditional machine learning models built with Random Forest and Gradient Boosting Ensemble techniques had a handful of models with very high performance in classification.

Nevertheless, they tended to be ‘black box’ models. While they could be scrutinized for feature importance, the rationale for the given answer was absent. This issue, like nearly every technological issue, created a multitude of problems in the security field where the explanation of a prediction made often stands on the same significance as the prediction itself.

1.4 The Advancement of Deep Learning Technology and Its Impact on Spam Filtering

The progress of a spam email filter provides scope for applying deep learning. In the context of AI, deep learning is a branch of AI that develops parallel with machine learning and, in this case, focuses on trying to imitate human perception and thinking. There is a new, ubiquitous pattern towards using deep predefined neural nets within software systems called "user-facing." It becomes easier to build applications that, because of the ever-increasing amounts of unsolicited spam emails, need to be able to respond to the need for spam filters.

Deep learning has an extraordinary and crucial feature: its capability for feature interaction. Unlike traditional methods of deep learning, which are limited to perceiving one target feature at a time, deep learning utilizes and boasts a much larger target feature.

Deep learning feature interactions are too deep learning what n-grams are to language modeling: subset combinations based on their given 'features.' A typical example of this is how deep nets learn the association between the frequency of occurrence of some words and strings of characters of emails that are often tagged as spam. Furthermore, deep learning approaches are flexibly combinable and alterable; under some circumstances, they can exploit noise and variability from real email data. The self-learned feature representations are complicated enough to justify using a hybrid approach to deep learning, although it is still reliable with some other well-defined one-dimensional models like MLP.

1.5 The Rationale for An Integrated Approach

Integrating approaches to spam filtering is beneficial because both contain exceptional advantages. For instance, tree-based ensemble techniques such as random forests and gradient boosting are well known for their proficiency in handling tabular data. They are accurate and also robust. However, deep learning models are well-known for capturing complex non-linear interactions between features and employing modern interpretable AI techniques like SHAP (Shapley Additive Explanations) to interpret the results.

The purpose of the hybrid model is to bring together distinct methodologies. In this instance, a deep learning model developed in Keras was incorporated into the framework to capture nuanced variable interactions. In addition, traditional ML methods provide a lower-bound performance guarantee, enhancing the framework's credibility. In addition, this hybrid framework with SHAP method is able to address the “why” questions of the predictions by uncovering the features responsible for an email, such as frequencies of words, special characters, capitalization, and others being classified as spam.

Spam continues to evolve in sophistication and volume, making this integration more crucial. While the system is able to handle the changes, almost equally important is that it can explain how it derives its decisions. By integrating both the differentiation and comprehension problems of modern spam filtration, the system becomes more effective and reliable.

1.6 Issues Relating to Trust and Security within Email Accounts.

Effective spam filters do more than block emails. They protect sensitive data, avert cyber strikes, and contribute to the preservation of Information and Communication Technology (ICT). The security of the systems in many organizations rely on the effectiveness of the spam filters, which makes the performance of these systems highly critical. Different stakeholders, from IT experts to non-specialist users, need reliable models explaining the system's mechanisms.

Unlike random patterns, users understand that plausible reasoning textual patterns are employed to make decisions, which fosters practical trust. The hybrid framework raises practical trust by deploying important textual features and explaining which signals drive spam classification. This is important for further developments. With the automation of spam filtering, security operators have the means to enhance the model, minimize biases, and better manage new spam detections. An explainable model also makes it possible for the balance between automation and manual intervention, enabling spam detection systems to be robust and accountable.

As mentioned before, spam is still evolving and, therefore, needs more sophisticated detection mechanisms. Detection systems have traditionally been ruled-based, but now, they have shifted to machine learning and, more recently, deep learning, which brings pros and cons. Problems like the gap in detection and the development of detection methods by spammers are way too sophisticated to be solved by a single method. These solutions require the trust and explainability of ensemble ML combined with the non-linear complex interaction capturing of deep learning, which is not ideal for one method to tackle.

1.7 Problem Statement

The problems in determining spam messages involve the distinction between spam and non-spam messages and the changing techniques of spam messages (Hemalatha et al., 2022). Hence, the problem of coming up with a hybrid deep and machine learning framework that applies explainability techniques to uncover important salient text patterns that combine prediction from the models with email security in the real world arises.

The statement above captures the core challenges currently faced by spam detection systems. Emails that are classified as legitimate often have elements of the language used to communicate, such as images, text, or hyperlinked materials, which are now associated with spam. Therefore, this phenomenon presents considerable difficulty in eliminating both false negatives and false positives, ultimately jeopardizing the credibility of the filtering system.

In addition, one reason spam detection is becoming more difficult is that spammers are constantly improving their techniques to avoid detection. They employ various forms of adversarial strategies to defeat traditional email filtering solutions. Keyword obfuscation and dynamic content manipulation filter out static rules that are easy to get around. Over time, these techniques become less effective because models that do not adapt to real-time changes in new patterns will become stale.

Another issue of concern is most existing models' “black box” features. Modern machine learning approaches, particularly ensemble methods such as Random Forests and Gradient Boosting, perform class separations well. However, they do not explain the major

contributing features involved in the classification. Such opacity is especially alarming for security-related domains, where some explanation is necessary to trust a decision, take responsibility for it, and improve the system.

The justification for a new solution in this case is double. First, there has to be a solution that maintains high detection effectiveness regardless of how ambiguous the content of emails is. Secondly, the solution must be able to explain the most important textual elements, such as frequencies of certain words, occurrences of certain characters, or patterns of capitalization. This sets the requirement for a hybrid framework that combines traditional and modern approaches to machine learning. The framework is expected to perform well and, in addition, provide explanations for its predictions.

In this case, the aim is to leverage the ensemble method's tree-based learning and deep learning implemented in Keras to capture both simple statistical correlations and complicated, non-linear interactions of features. Such a system would be able to provide high rates of spam classification while remaining explainable to users within the email security domain. Essentially, the problem is not only about high precision performance. A trustworthy model is one where the user can see what features of the text determine spam and non-spam messages.

The entire study is defined by this need for an interpretable, over-adaptable solution that performs at its peak. Then, it outlines the design of the hybrid framework, followed by methodological and experimental work. This ensures that the suggested framework deals with sophisticated spam techniques while still being explicit in its reasoning and satisfying the practical needs of today's email security systems.

1.8 Thesis Statement

An integrated machine and deep learning framework and model that uncovers important textual attributes associated with spam classification provides a solution for spam detection, which improves effectiveness and robustness against changing spam manipulations.

At the center of this research is the claim that a single, fixed modeling approach is not suited enough to solve the problem of spam detection, which has several dimensions.

Traditional machine learning techniques like Random Forests and Gradient Boosting have dominated the Spambase dataset because of their efficacy in processing tabular data.

Their modeling of non-linear interactions using ensemble techniques has attained outstanding performance and robustness. These semi-automated techniques earned great applause, yet they neglected a very salient issue of the interpretability of their results. The classic 'black box' problem makes it difficult to explain basic questions like why the email was classified as spam.

In terms of learning complex, nonlinear feature interactions, Keras' deep learning methods are far superior to the rest. A good example would be the more traditional algorithms failing to discern the details within the text, while a deep feedforward network can identify these patterns. However, many deep learning structures initially intended to tackle sequential or spatial data, such as CNNs or RNNs, pose problems when applied to spam detection, a static and tabular domain.

This thesis argues that merging these approaches, ensemble machine learning and deep learning, makes spam detection systems not only more transparent but also much more effective. A proposed hybrid model is meant to improve both these processes by providing features from both and utilizing their strengths. Using tree-based models offers enhanced performance with ease of interpretability. In addition, the Keras-based deep learning model will automatically detect complex interactions among features and delicate textual signals that other models usually ignore. What stands out in particular about this hybrid model is its emphasis on transparency.

With explainability frameworks like SHAP, the system does not merely forecast a spam email but also indicates which specific features—word frequency, character sequences, or a capitalization ratio—had the most impact on the decision. This understanding is essential for users and security experts who rely on these systems to protect sensitive data and build trust in automation. In addition, the framework has been created to withstand changes in spam strategies.

Since spammers constantly modify their approaches, it is essential to provide high classification effectiveness while revealing decision-making processes. The hybrid model guarantees that if some features become poor predictors of spam because of new strategies, the model as a whole will still perform, satisfying system requirements for robustness and updatability provided by the interpretability component.

The thesis rests on the premise that current spam detection technology requires the development of a hybrid machine-deep learning model that is predictive and explainable.

The construction clearly allows for incremental performance improvement while guaranteeing that the classification is performed in a manner that can be understood, thus fulfilling both practical and technical challenges in email security.

1.9 Research Objectives

This study aims to create a spam detection system with optimum classification performance and model interpretability. Some systems automate spam detection and have many infesting spam techniques. The models use output-only prediction methods and, therefore, do not consider system internals. A substantial gap needs to be filled, which assists the user in understanding what parts of the text invoke the prediction. The preliminary aim, enhanced model interpretability, relates to fusion techniques in the filters that use SHAP in the check part of the system. It is not only an explanation system but rather a system that considers the outcomes provided for each feature. Moreover, the system is designed to provide adequate interaction and response to the so-called “black box” systems, which are systems with complex algorithms.

Flexibility is an essential feature of this research. New types of spam messages require detection and processing within a relatively short time frame. Improving adaptability is a clear manifestation of the system having the ability to locate and rank pertinent textual signals in the proposed framework in the most optimal way possible.

This could be achieved by integrating classical approaches of machine learning, which excel in table-like structures, with non-linear deep learning models that are adept at complex proficiency interactions. These strategies should help establish a system that

keeps operating even when advancing spam attempts are made to reduce the effectiveness of filtering solutions.

The research's performance objectives aim to go beyond anticipated levels to achieve purposive output. When stakeholders are provided with clear descriptions of what is happening within the model, for example, the character or word frequencies that comprise spam alerts, there is a higher probability of endorsing the model’s results for operational use.

This paper examines the application of the hybrid methods. Low-level machine learning approaches, like Random Forests and Gradient Boosting, have always worked well with algorithmically organized data, including the Spambase dataset. Unfortunately, these models generally break down with intricate feature interactions. Deep learning techniques, particularly with Keras, are utilized primarily on non-structured data because they correspond with profoundly embedded non-linear relationships. The current work intends to integrate the two paradigms to circumvent the constraints of both working methods. It is expected that the precision classification and the prediction logic will improve with these methods.

The analysis incorporated in this report is most certainly essential considering the removal of features in feature resilience analysis. The research attempts to determine whether the model relies on a dangerously small set of signals by removing the top-performing features identified by the explanation techniques and observing how the performance changes. A slight dip in performance within these tests would indicate that the model is flexible and not deeply dependent on a specific dominant cue. In this regard, the

study assures that the hybrid framework is not only efficient but also, as it emerges under duress with changes in spam strategy, proving its efficacy.

Evaluate a feature with resilience and consider their associated explanations as tests need to be conducted and further work needs to be accomplished. A feature or class of features known to operate with a small number of central signals may not be easily accommodated. Furthermore, a model that tracks the "X" feature's activity and employs data reduction methods to focus on feature "X" does not necessarily indicate that this is the best approach and will lead to improved performance. Unproductive features may also be pruned to improve performance, which will lower functional and analytic capability.

These results suggest a non-high flexible window that can sustain system performance but allow for some degradation due to the shifting spam strategy. There also seems to be a strategy that targets optimal delineation, yielding a more than intelligent increase in system efficiency and changes in spam strategy.

1.10 Research Questions and Hypotheses

1.10.1 Research Questions:

- **RQ1**: Do hybrid models integrating machine learning and deep learning yield robust spam detection while highlighting key textual features?
- **RQ2**: Will combining diverse textual signals yield broader coverage of spam patterns and maintain strong performance?
- **RQ3**: Is the classification outcome significantly affected when selected key features are removed?

1.10.2 Research Hypotheses:

- **H1**: Hybrid models will achieve an F1-score above 0.90 and effectively expose the key textual patterns that drive spam classification.
- **H2**: Integrating diverse textual features will enhance the detection performance by capturing a wider range of spam indicators.
- **H3**: Removing the top textual features will only modestly decrease performance, demonstrating the model's resilience and ability to avoid over-reliance on any single signal.

The research questions and supporting hypotheses aim to address the performative and interpretive aspects of a blended spam detection problem. RQ1 and H1 assess the effectiveness of traditional machine learning and deep learning unit integration in a spam detection system. This research aims to validate a model that predicts an F1 score above 0.90, thereby raising the benchmarks of reliability and accuracy in detection. Also, the focus on significant feature text disclosure assures that the model is not a black box but rather a model that explains why emails are classified as spam. This quality is vital in developing systems that greatly automate sophisticated tasks, such as email filtering systems, which need to work and be explained.

H2 and RQ2 pursue this line of research by investigating the advantages associated with the aggregation of multiple textual signals. A spam email's design, content, and structure can vary significantly. The study suggests that combining features such as word count, character count, and capitalization will enable the model to employ a broader range of spam manipulations. This approach will most likely maintain high spam model

performance because the feature integration is expected to incorporate more spoofing meta features than a single feature or small range of features would provide. Ultimately, RQ3 and H3 study the model's robustness by removing some key features. The model's performance based on these two signals refers to too much reliance.

The expectation that performance with and without these features would not differ greatly indicates that the model functions with a more distributed set of cues instead of relying singularly on the unitary dependency system. This aspect is important in real-life situations when certain features may shift due to some spam alterations.

1.11 Implications and Significance

1.11.1 Contributions to Knowledge

Integrating different modeling approaches for predictive analytics spam detection has practical relevance and contributes to the proposed hybrid framework. Primarily, the framework demonstrates that a system combining traditional machine learning methods and deep learning is superior to any of the components suffering from limitations.

Traditionally, machine learning models such as Random Forests and Gradient Boosting have been favored due to their good performance with tabular data and their simplicity of interpretability in feature importance. However, they often fail to capture the subtlety of many features' non-linear interactions. The hybrid approach captures the power of deep neural networks to model complex, multi-layered relationships utilizing the integration of a deep learning component developed in the Keras framework. This

transformation enhances performance and further contests the claim of the divide between ‘black box’ models and ‘white box’ interpretable systems.

In addition, the latest explainability techniques, such as SHAP (Shapley Additive Explanations), shift the discussion forward by assessing feature contribution with unmatched granularity. The model demonstrates only a modest decrease in performance when critical features are absent, suggesting the model possesses a strong capture of the textual patterns, which is even more perplexing. Such knowledge improves the understanding of feature interdependencies in high-dimensional tabular data and defines a scope for developing robust models for adversarial manipulations. Lastly, the hybrid model attempts to suggest a compromise between the predictive performance and the explainability of a model, particularly regarding complex classification tasks.

1.11.2 Practical Implications

These systems for automated spam detection can greatly benefit from the research outcomes. As to existing challenges in email security, perhaps the most problematic issue is the lack of explainability in the decision-making process. Many organizations employ spam filters that are algorithmically deemed valid yet fail to provide adequate explanations for their classification decisions. Such blindness can lead to mistrust, misinterpretation of results, or difficulties in system safeguarding. The hybrid framework defines key text features (e.g., word frequencies, character patterns, capitalization), which explain why some emails are classified as spam. Because the explainability of results is enhanced, security specialists and users can trust that the system delivers when too many false positive or negative results are presented. Thus, more action can be taken.

Furthermore, the hybrid approach has been designed to be adaptable. Each spamming method has its unique way of evolving, and static filters often do not have the means to adapt to these changes. Because the framework is directed at performance and interpretability, it is easily adaptable with the rise of new text models. For instance, if the explainability analysis indicates that some features are becoming less indicative of spam, the model can be altered without redesigning the filtering system. In addition, the system's perpetual k-fold cross-validation and multiple performance metrics assure efficiency as time progresses. This level of practical robustness is critical for organizations that seek to safeguard sensitive information and enable seamless communication, as it helps detect spam scams.

In this context, integrating deep learning elements is advantageous as it can pull out feature interactions that are too complex for some models. Even with the potential shortcomings of deep models, which include high computation costs and overfitting, the model built with an agenized architecture with dropouts and batch normalizations is guaranteed to perform and be efficient. This framework is suited for future use when there are moderate-size datasets, such as the Spambase dataset or large, diverse ones that need more complicated solutions.

1.11.3 Broader Impact and Future Directions

The hybrid framework is helpful in other forms beyond spam filtering, especially in areas requiring results that are both reliable and well-defined. Stakeholders in finance, healthcare, and even cybersecurity are increasingly concerned with model performance, and the explanation provided behind the output research, especially the incorporation of

classical machine learning, deep learning, and advanced-level interpretability, is a contribution toward the development of a general framework that applies to different and complex fields. The ability to assign responsibility for a prediction to specific pieces of text may, in a novel fashion, transform what is required in evaluating models and, in turn, how models are updated and enhanced over time.

In addition, the feature removal test evaluating model resilience is sure to achieve something remarkable in the future. The assessment that removing a feature is causing a performance drop indicates a much stronger dependency on textual signals and evaluates metric robustness in a highly defying manner. Other work will likely follow this line of thought by creating hybrid models, utilizing interpretability methods other than SHAP, or extending the framework to real-world databases outside the Spambase corpus.

The development of the hybrid model is of critical relevance due to the fast-paced technological changes. Protection against spam and other forms of malicious behavior requires increasingly sophisticated approaches. The model presented here does not only set high demands with regard to contemporary performance but expects that other systems will also be built around it. This ensures relevance regardless of the changes in the strengthening spammer tactics by providing the balance between interpretability and accurate detection.

The ability to generalize the results to other domains, coupled with this, shows how significant this research is. The study's results will also be important from a policy and operational perspective. For example, email-based organizations could leverage the design of auditable and effective spam filters. These filters enhance pre-compliance transparency and enable security personnel to modify relevant features. In this way, the hybrid

framework facilitates academic work and responds to practical issues in communication security.

As a closing remark, this model solves the contradictory demands of high performance and effortless decision-making in spam detection systems. This will have a broader influence because it will contribute to the other domains and may serve as a framework for constructing complex but interpretable predictive systems in sensitive environments. These results will provide a foundation for further research to enhance such systems' robustness, scalability, and ease of interpretation to eliminate undesired or untrustworthy outcomes.

Chapter 2: Literature Review

2.1 An Outline of Different Strategies Employed for Spam Detection

Spam Emails, often referred to as junk mail, are among the most harmful types of communication sent through email channels and continue to pose a challenge for most, if not all, users today. McCuster noted that approximately 65 percent of emails sent outfall into the category of spam emails (McCuster,2005), which indicates how frequently unwanted or unsecured communications are sent.

In order to keep pace with the increase in unsolicited emails and advanced cyber threats, spam detection systems have evolved in tandem with the advancement of email communication. Other methods, such as Heuristic and rule-based, are considered outdated in contemporary spam classification systems. For instance, heuristics such as expressions, keywords, and even blocklisting were popularly used. For instance, emails containing phrases such as “free money” were automatically routed to the junk folders (Sahami et al.,1998). These techniques, however, provided low levels of security for holistic systems since spammers began employing diverse bypass methods to these filters.

The classification performance increased significantly with the adoption of new algorithms, including machine learning (ML) and other statistical approaches. For instance, the Naive Bayes model is an ML model that applies reasoning, such as the likelihood of certain words being linked to spam emails. McCallum and Nigam (1998) demonstrated that Naive Bayes achieved high performance on large email datasets, even with feature independence assumptions. This provided more latitude than the rigid systems, which were based on predefined filtering rules, and these systems were far too limiting.

While applying decision trees brought about new forms of advancement in spam filtering, the ensemble methods made the most impact. Random Forests, which Breiman (2001) built on the idea of ensembles of decision trees created from random samples of data and random subsets of features, was a major difference. Model accuracy increased while overfitting decreased, leading to more precise results. This was further advanced by Friedman (2001) with the introduction of Gradient Boosting Machines (GBM), which utilized sequential training of models to mitigate the effects of trained models.

Controversial reviews of ensemble methods were provided in 2009 by Guzella and Caminhas, who stated that these strategies had outdone the single classifiers level, especially while filtering spam in tabular datasets (Guzella et al., 2009). In addition to the aforementioned probabilistic and tree-based models, support vector machines (SVM) have also been used in spam filtering. For example, Drucker, Wu, and Vapnik (1999) observed that class margin maximization, arguably the most characteristic feature of SVMs, makes it particularly useful in segregating an email from Spam. The logic makes sense in principle, but the SVMs are incredibly expensive to compute and depend on optimized parameters, so only end-scale systems can use them (Drucker et al., 1999).

Spam's sophisticated attributes have complicated patterns within data, which must be understood as the tools utilized to detect spam need more enhancements. This initiated change brought up the effort to try to apply deep learning methods. Deep feedforward networks such as Keras (Chollet, 2015) can understand intricate non-linear interactions between features and multi-faceted powerful features of the data presented to them.

Although deep learning has progressed tremendously in computer vision, natural language processing, and so on (Goodfellow et al., 2016), it is so far optimistic and

challenging regarding its use in spam detection. One of the more significant challenges is that they are concerned with feature-extracted datasets (like word frequency, character frequency, and so forth) instead of raw sequential or spatial data. Consequently, deep learning models that are multi-layer perceptions (MLPs) tend to perform satisfactorily (Zhang et al., 2018). However, these models are quite problematic because of their lack of interpretability and the extensive efforts needed to minimize overfitting, that is because of their non-intuitive nature and different perceptions of the model.

The literature suggests that throughout the years, there has been an emphasis on analyzing the model's decisions rather than simply achieving performance. A number of scholars argue that adapting to dynamic spam strategies and earning users' trust is a function of understanding the underlying causes of spam classification. Hence, attempts were made to incorporate SHAP and other explainable features into spam detection models. This is useful for explaining how hostile filter modifications can be made (Zhang et al., 2018).

The broad application of machine learning and deep learning techniques enhances the effectiveness and efficiency of spam detection algorithmic processing far more than the use of traditional rule filters. Many of the previously used spam filters lacked generalization, which made the existence of hybrid approaches to data processing, including machine learning and deep learning, imperative. Each of these approaches allows for the generation of solutions in response to the growing problems that need to be addressed in spam detection.

2.2 Machine Learning Approaches to Spam Detection

Despite incorporating ensemble strategies, spam detection systems have applied probabilistic techniques for many years. These methods, initially implemented to solve the challenges posed by rule-based filters, have since evolved to accommodate numerous sophisticated tricks in complex email databases.

The Naive Bayes classifier has been the focus of attention of several researchers aiming to deal with the spam issue due to its ease of handling and effective outcomes. Numerous studies have applied algorithms and successfully achieved high performance (Ghosh et al., 2020).

Extensive research has already been done on plain-text emails, and the results were in high detection rates (Deshpande et al., 2007). Based on the relative frequencies of tokens in their corpus, Deshpande and Harris (2007) constructed a token filter and trained it using a huge corpus of spam and legitimate messages. Then, they evaluated it on newly received personal messages. Different threshold values were tested to determine the optimal configuration for the anti-spam filter, with the cutoff set as the cost-sensitive Bayesian anti-spam filter. More conservative measures had to be taken to deploy a Bayesian anti-spam filter. Still, it could function as a first-pass filter with great effectiveness (Deshpande et al., 2007).

A spam filter efficacy can be measured by how accurately Spam is Spam that goes through it. This efficacy also has other measures of performance. This concern stems from applying some smoothing method that attempts to change the estimate of a conditioned event because of the phenomenon known as data sparsity. In their paper, Gurneet Kaur and

Neelam Oberai (2014) tried to improve the performance of the Naïve Bayes Classifier in classifying Spam emails by suggesting modification of the Jelinek-Mercer Smoothing and Dirichlet Smoothing method to be used instead of Laplace method of Naïve Bayes Classifier. Therefore, the Naïve Bayes Classifier is modified where the smoothing techniques employed are designed to estimate the collection probability of the corpus for the model. The modified technique demonstrates impressive performance when applied to large datasets with particular specified keywords and multiple smoothing factors. The modified smoothing technique uses uniform distribution probability to estimate the collection probability. The enhanced technique gave the optimum outcome in shifting keywords with datasets and varying smoothing factors (Kaur et al., 2014).

In a detailed analysis, Androutsopoulos (2000) scrutinized Bayesian naive filters and keyword-based techniques and noted a general superiority of ensemble methods (Androutsopoulos et al., 2000). Additionally, Rokach published a comprehensive review of ensemble classifiers in 2010, demonstrating that the reliability of a classification improves remarkably with the use of multiple models, beyond simply achieving correct classifications (Rokach, 2010).

Although Naive Bayesian filters are optimistic, further research shows that more safeguards will be essential for their implementation (Deshpande et al., 2007). In any case, Naive Bayesian classifiers are still pertinent and useful for spam detection.

Due to advanced spam tactics, the one-model approach has become less efficient with the variation of emails. Some specialists argue that the intuitive nature of splitting features through threshold rules makes individual decision trees a viable alternative to probabilistic models. Unfortunately, single-decision trees have well-documented fragility

problems and overfitting in high-dimensional and noisy feature spaces. The increasing popularity of ensemble classifiers has somewhat overcome these deficiencies.

The Breiman Ensemble method is famous for featuring a Random Forest model, which, like everything else, became prominent over a decade ago. Random Forests try to solve the overfitting problem in a model by taking an average of all the predictions of a multitude of decision trees, each trained to different subsets of the model's data. On top of this, the Random Forest algorithm allows the user to compute the impact of numerous variables in the model, which is helpful in spam classification, depending on the text attributes, such as frequency of words and punctuation marks.

The use of decision trees for spam email detection is not new and has been done many times, often in combination with other approaches to achieve better results. Several studies have reported some encouraging results using such combined methods. For example, one achieved 91.67% accuracy by combining decision trees with logistic regression, and another integrated harmony search algorithm with decision trees and achieved 95.25% accuracy (Zekriyapanah et al., 2017). Also, decision tree approaches are known to the authors to have high accuracy rates. According to Pandey (2011), the Decision tree classifier trained with the relevant attributes of spam emails provided more than 86.74% accuracy and up to 95.25% across various studies (Pandey et al., 2011).

Certainly, there are many characteristics in the spam email categorization, and one or more features are associated that influence the identification and categorization of spam emails of no importance. Defining attributes is important in spam email filtering systems. Important features can be selected that improve the classification performance.

A feature selector creates a feature subset such that the predictions of the target variable can be produced with reasonably high certainty. Feature selection and classification were done using the Harmony Search Algorithm and decision tree (Zekriyapanah et al., 2017).

The results received from Spam- the spam-based dataset is reported in the proposed model; the postulated recognition accuracy was recorded at 95.25 percent. This figure is high compared to SVM, NB, J48, and MLP models. The model also performed satisfactorily compared to the proposed manual markup models with recognition rate in collapsing classification scopes, especially with NB, SVM, and LR models. When the Ling-spam and PU1 datasets were analyzed, the accuracy percentage for recognition was considerably higher (Zekriyapanah et al., 2017).

Shweta Rajput and Amita Arora, in their paper, thoroughly analyzed a set of classifiers, both pruned and unpruned, using the Weka tool with the Spambase dataset (Rajput et al., 2013). The synthesis of the accuracy results follows the criteria specified earlier. Unpruned data achieved an accuracy of 93.26%, while pruned data had an accuracy of 93.28% using 10-fold cross-validation. With the percentage split measure, unpruned data achieved an accuracy of 92.26%, and pruned data achieved 92.90%.

Moreover, the results of pruned trees indicated that their performanceis better than that of unpruned trees. In the case of Grafting, the results were even better. Grafting is a heuristic strategy for creating new decision tree nodes. It was observed in Rajput's study that J48graft is the best performer when pruning and Grafting are done on a single dataset.

Pruned trees took moderate time to build the model with decent accuracy. On the other hand, the Simple Cart model was constructed in very little time, but its accuracy suffered due to too much pruning. J48graft outperformed other algorithms in predictive accuracy (Rajput et al., 2013). The pruning technique uses only the available information as the tree grows, while the Grafting technique enhances a Decision Tree's efficiency.

Pruning differs from Grafting as it tries to lower the decision tree's complexity to an acceptable level of prediction performance. Also, Grafting adds complexity to the tree. It was shown that pruning and Grafting, which are opposites, can work well together.

Therefore, it was shown that a combination of pruning and Grafting in a decision tree outperforms the two methods individually on a Spambase dataset. Therefore, applying decision trees in classifying emails is a process of accurately and automatically mining the spam patterns from the spam base dataset to build a spam model that can be used in any computing environment (Rajput et al., 2013).

In addition, there are new, exceptional ensemble procedures called boosting methods. Freeman and Schapire implemented the first version of this concept with the establishment of AdaBoost, one of the algorithms in ensemble learning that combines weak classifiers into a strong classifier using weighted voting and robust features. It selects and gives weight to various classifiers to obtain the best possible outcome (Ding et al., 2022).

Boosting improves a model's performance by enabling it to learn from its most unsuccessful attempts (Stamp, 2017). Boosting algorithms have been successful in spam filtering. The Random Boost algorithm, a combination of Logit Boost and Random Forests,

was more efficient and less computationally intensive than previous methods (DeBarr et al., 2012).

Active learning seeks to reduce the number of labeled examples needed to create a reasonable classifier, which is useful in spam filtering. Nikhila studied the effectiveness of active learning by implementing a confidence-based label sampling approach with Boosting. AdaBoost was implemented to train a classifier and create a scoring rule that allows the active learning strategy to choose the most informative samples for labeling (Nikhila, 2008).

Also, Bhardwaj, in their study, claims that ensemble methods that combine bagging and boosting techniques are more accurate than individual classifiers. This paper attempts to develop an email spam detection system using ensemble classification approaches based on bagging and boosting machine learning techniques. The Ling-Spam Corpus was selected as the dataset for the experimentation and testing. The spam detection is performed by bagging the MNB and J48 decision tree classifiers and then applying the Adaboost algorithm to enhance the performance of the classifiers by converting weak learners to strong learners. The experiments set forth here consist of three distinct tests, each producing results that can be analyzed comparatively. The first set of tests combines the single classifiers, the second uses bagging, and the last uses boosting to detect spam emails.

Ensemble techniques were applied to single classifiers, and the results were compared and analyzed using defined evaluation metrics (Bhardwaj et al., 2023). Support vector machines, along with other classifiers, have been utilized in spam filtering, along with boost and tree ensembles. Joachims (1998) paid attention to the margin that was achieved by the classes and hypothesized that SVMs may work in higher-dimensional

space. Many of these SVMs, however, are not practical for large-scale real-time systems because they tend to be exceedingly costly in computations and very fragile in relation to the perturbations of the SVM's parameter values (Joachims, 1998).

In an attempt to improve the performance of SVMs, researchers have suggested several improvements, such as implementing a feature selection method that uses genetic algorithms (Wang et al., 2005). This study aims to achieve GAFS or Gradient Aerogel Fibers in SVM email classification systems. The integration of GAFS improves the performance of the SVM classifier, thus reinforcing the GA-SVM. In GA, the scaling factor, which represents the value of the feature for the specific classification problem, is calculated. A strong biasing operator is used to encourage sparsity among the scaling factors of the features. Through a set value of the threshold, the irrelevant feature's specific scaling factor is eliminated. The results obtained with the UCI Spam repository database validated that GA-SVM is superior to the initial SVM classifier as it provides better classification results with a small number of support vectors (Wang et al., 2005).

Reddy and his colleagues investigated the performance of a Support Vector Machine in classifying emails as Spam or ham, with a particular interest in Spam text classification. Their sample consisted of 5572 emails, 747 of which were Spam and 4825 of which were ham. Once all data collection, preprocessing, and balancing were completed, they obtained a 95.32 percent accuracy score. Following that, the Support vector machine algorithm was applied to classify the messages into two classes: spam and ham (Reddy et al., 2021).

Class imbalance and feature augmentation are still problems within traditional ML models. In spam filtering, text filtering has been researched in many studies, such as those

by Forman (2003), who has done much work on feature selection. Seiffert, through his study conducted in 2010, describes RUSBoost as an improvement of the build and improve paradigm, which involves random under-sampling of the majority class during the boosting iterations. This addresses the class imbalance problem in spam detection (Seiffert et al., 2010).

To utilize traditional techniques, an example referred to is spam filtering. The performance was significantly improved using decision tree filters, ensemble methods, and even probabilistic model estimators like the Naive Bayes. While these models were understandable, they were too dependent on oversimplified static features devoid of complex interaction. Most of the literature ignores these kinds of gaps in the frameworks.

In this case, there is a reason to try to bridge the gap between deep learning and the traditional approaches to machine learning, which will be discussed in the next chapters.

2.3 Approaches to Spam Detection through Deep Learning

The detection of spam involves automatic learning of the most relevant features from the raw data, which is why the task is challenging. A spammer that tries to avoid conventional rule-based filters is the hardest to deal with. Techniques in deep learning, such as convolutional and recurrent neural networks (CNNs and RNNs, respectively), have been known to fully exploit the powerful automated feature extraction capabilities of CNNs and RNNs, and are known to be able to overcome these obstacles (Shoba et al., 2023).

Srinivasan (2020) argues that deep learning models perform better than classical models in machine learning spam classification, especially when combined with embedding methods. The experiments indicate that the deep learning architecture

outperformed the standard machine learning classifiers on accuracy, F1 score, precision, and recall. This is important because deep learning architecture can learn the hierarchical, abstract, and sequential feature representations of emails.

Additionally, deep learning word embedding is superior to traditional email representation techniques. Incorporating the email's syntactic, semantic, and contextual similarities into simpler forms is what enables word embedding. Thus, deep learning combined with word embedding can be used in real-life situations of spam email filtering (Srinivasan et al., 2020).

In a different research work, Shodh and others (2024) suggest that the feature selection N-gram TF.IDF can be combined with spam detection deep multi-layer perceptions -neural network and improved by a distribution-based balancing algorithm.

The method was tested with the SMS spam collection and social media datasets, which reported greater performance than the industry standard due to high-dimensional data and class imbalance challenges. Furthermore, because it incorporates complex features that yield very low false positive and negative rates, it greatly improves the classification of spam. These outcomes favor the statement that deep learning coupled with intelligent feature extraction and balancing methods is an adequate answer to spam detection.

Sheneamer (2021) has also proposed a technique that combines CNN and LSTM models, with and without the unsupervised learning algorithm GloVe model integration, to classify messages into spam and non-spam categories. Sheneamer uses only email data; no human involvement is required at the feature extraction stage. This research analyzes the effectiveness of classical machine learning algorithms compared to deep learning

techniques on spam collections and intrusion detection systems. That research proved that deep learning techniques had the highest measured values of precision, recall, and accuracy. Unlike other machine learning analysts, Sheneamer employed a variety of approaches to deep learning and demonstrated the feasibility of advanced email spam filtering. The most incredible accuracy of 96.52% was achieved when applying CNN with Glove to a benchmark dataset of 5,243 spam SMS and 16,872 non-spam SMS messages (Sheneamer, 2021).

Multi-layer perceptions (MLPs) are an example of an artificial neural network model known as a deep feedforward neural network. Multi-layered structures consisting of two or more layers, or deep learning neural networks, categorize data linearly using activation functions such as ReLU or Rectified Linear Unit for hidden layers and sigmoid function for the output layers where binary classifications are made. MLPs, for example, regarding spam filtering on Spambase datasets whose feature set consists of word and character frequencies and capitalization, can understand the relationships among the features without any pre-processing steps. Regularization, such as dropout or batch normalization, enables the MLP networks to be relatively cost-effective while achieving reasonable performance (Zhang et al., 2015). Also, Peters, in a study published in 2018, demonstrates that deep contextualized representations perform better for text classification even in non-convolutional networks, and MLPs have also been known to use them, thus broadening the range of applications MLPs using pre-trained embedding (Peters, 2018). Text categorization benefits from the use of CNNs developed for image processing.

Since n-grams can be extracted without feature engineering, CNNs can learn local structures by performing convolutional scans over inputs. Even after Kim, in their study in

2014, demonstrated that a fundamental configuration of CNN could achieve state-of-the-art results in sentence classification. The results have been applied to spam filtering via spatially constrained textual features. For spam filtering, CNNs can operate on raw textual data without processing (words or letters) or supplied text features (Kim, 2014).

By capturing certain patterns, like conversational phrases and stylistic features, with learned convolutional filters, CNNs reduce the effort needed in the construction of features by labeling each example. Yin also has reviewed other works that argue that CNNs can surpass some traditional models in text classification when the model design is adapted to a particular language (Yin et al., 2017).

Researchers have improved Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) models for sequential data types. For instance, LSTMs are useful with spam detection, where the email body is the input due to their capability of capturing the temporal structure in the sequences of words. LSTMs are considered a more sophisticated version of RNNs because, unlike other RNNs, they have gates that control the flow of information within the sequence and are, therefore, able to avoid the vanishing gradient problem (Hochreiter et al., 1997). Thus, LSTMs can utilize contextual information and the semantic relations among spam words for most complex spam messages. The work of Yin and others also supports the findings of Pennington and others in 2014, who introduced GloVe embeddings in claiming that, besides proper word forms, LSTM structures are very competent in text categorization (Yin et al., 2017).

Nonetheless, for those working with LSTM on Spambase, the usefulness of sequential dependency modeling may not be as significant due to the dataset being externally feature-engineered.

An example of Natural Language Processing (NLP) deep learning is the development of self-attention mechanisms and the BERT-based architecture or Bidirectional Encoder Representations from Transformers. Self-attention simultaneously captures global dependencies of the text; it permits the yielding of context representations. Devlin and others (2019) reported that the BERT model is among the foremost impacts in Natural Language Processing through deep bidirectional transformers. Such complex systems can detect spam but require enormous amounts of computing power and training data. Newer, more sophisticated systems like Distill BERT are appearing (Devlin et al., 2019). The application of transformer models with neural networks to traditional tabular datasets, such as Spambase, remains an unsolved problem; however, there is no doubt that the richness of captured patterns of marked text will be outstanding.

As monolithic hybrid deep learning frameworks are incapable of solving every issue at the same time, these frameworks need to be divided into various architectural models. For instance, hybrids may integrate CNN-based feature extraction and LSTM sequence modeling or combine conventional feedforward wiener networks with attentive feedforward versions. This was shown by Johnson and Zhang (2015) in their development of a text classification system where convolution and recurrent layers were used for feature extraction. In the context of spam filtering, some hybrid multi-deep learning models are known to outperform many single monolithic systems. These systems will perform well with unstructured and structured data since all available information is valuable (Johnson et al., 2015). The boost of feature extraction processes automates what is already manual, creating new deep learning obstacles within the scope of feature engineering or extending old ones.

Currently, researchers are attempting to work on outer processes to solve deeper issues. However, the amount of computation, or hyperparameter tuning required, is, at best, inconceivable and astonishing, which is mutually valid for deep models. The AI is severely understudied in spam detection because the problem is that deep learning lacks shallow incoming data. For example, the Spambase dataset is well-known for its “crafted” numeric features and offers the deepest reward for deploying deep feedforward neural networks.

Conversely, some models, which are a combination of CNN and LSTM, are better at detecting the content of emails. These models are referred to as Long Short-Term Memory networks.

The divide between explainability, effectiveness, and model complexity hinders using deep learning systems for efficient spam detection automation. Automating engineering workflows via deep learning for sophisticated spam filters is achievable with traditional feedforward, convolutional, and recurrent neural networks and transformers. However, no single integrative framework incorporates all of these methods simultaneously. This limits the traditional forms of AI and machine learning that prevail in automating processes within autonomous systems and the organization of system functions and processes.

2.4 Interpretability and Explainability in Spam Detection

Understanding contemporary methods of the taxonomy of spam text messages is all but straightforward. Even the most skilled and beginner learners can find the concept of classification particularly challenging, as is the case with deep learning models. Model explainability is still a major challenge in many domains, such as email filtering.

Furthermore, it is necessary, especially during system improvement, and user acceptance evaluation. Applying deep learning models creates a boundary that seems impossible to define in the context of spam detection: deep learning models operate like a black box. Deep neural networks operate on a 'fully connected' architecture, losing all forms of reasoning and decision-making due to the multitude of layers. One cannot trust deep learning models unless the actual discriminative textual features of the predictions are known, and in this case, the features that led to the incorrect predictions are absent.

The latest research concentrated on the issues brought forward by explainable AI (XAI) involving LIME, SHAP, and other XAI techniques. For instance, Lundberg and Lee (2017) are unabashed admirers of SHAPs, specifically, Shapley Additive Explanations, who strongly argue that it serves the purpose of interpretation and has explanatory value concerning the phenomena generated by the model in lieu of the features employed by the model. They went as far as saying that the frequencies of words, memorable characters, and capitalization were, in fact, constructive and explanatory for the workings of the spam filter and its outcomes (Lundberg et al., 2017). This subset is critical because it provides an explanation that, in simple terms, captures the model's dependence. In other words, the decisional competence expectation is an output of an actual domain reasonable expectation.

A model guessing a positive outcome is made by initiating an alarm, which will be checked by the analysts later. In Schrimpf's (2020) case, an analyst who has to analyze the SHAP Explanations will be alarmed. This is useful from a customer service perspective, as it reduces the false alarm rate in an ML pipeline, but tough from an engineering angle (Schrimpf, 2020). Consequentially, the use of SHAP Explanations to aid the manual analyses that need to be performed is being made more frequently by the Explainable AI

field. It is rather convenient for analysts to use SHAP explanations to detect false positives, which is why a study by Yusuf and others (2022) suggested SHAP Explanations would automate reducing the incidence of false positive results.

They incorporated ground truth labels in clustering and rules detection metrics to evaluate the efficacy of SHAP Explanations in resolving the problem of false positive versus true positive classification. Their work substantiates that SHAP Explanations are pertinent to the samples and can automate as many as 68 machine learning processes whose main objective is to minimize the generation of false positive outcomes (Yusuf et al., 2022).

Some support could be provided for AI applications and compliance to aid interpretation. There is a constant interpretation of model output in the finance, fraud, or healthcare domain because of the great demand (Doshi-Velez et al., 2017). Although that type of logic may not hold for spam detection, there is much greater performance, fairness, and controllability in a system that issues predictions that can sufficiently be interrogated.

A validation exercise should verify if the feature associated with Spam was truly a spam feature or if it was just a result of the training. There is a remarkable self-explanation: the head system of models has a self-contained structural performance supervision deep inside. Explaining and evaluating the cycle of features of a subsystem contributes to the system's transparency and to its robustness to negative impacts inflicted on it because of changes that may occur to the spam processes.

Alaoui and others (2024) state that incorporating Explainable Artificial Intelligence (XAI), especially SHAP, into spam detection models improves their performance,

transparency, and trustworthiness. This is one-way XAI has been used to augment spam detection systems (Alaoui et al., 2024).

2.5 Research Gaps and Future Directions

These gaps will be filled by looking at spam detection from a different angle. Li and others in their study (2022) have heavily relied on mark extraction techniques such as TF-IDF and Bag-of-Words (BoW) for spam classification and have shown that sequences of capital letters and a high frequency of unique characters are good indicators of spam (Li et al., 2022). Their approach does not stop identifying external characteristics of spam. It is, however, more than what their model has been evaluated for. Unfortunately, feature dependency evaluation has not been conducted, which poses the risk of overfitting.

Therefore, this study proposes doing feature elimination tests to see if the model has overly depended on specific features. This study aims to address this gap by assessing the extent of performance degradation inflicted by removing certain features to ensure effectiveness against the manipulation or alteration of spam signals.

Another significant weakness in studies such as Osman and Aljahdali (2017) is the lack of feature interpretability of the trained machine learning models. While the study reported high-performance results for both hybrid and deep learning models, there was no mention of which features were the most critical in spam detection. In this particular study, however, an attempt will be made to use SHAP (Shapley Additive Explanations) so that critical features for spam detection can be automated along with explaining the model's outcomes, thereby generating trust, and understanding of the spam filtering process (Osman et al., 2017). Trustworthy explanations of spam detection and the corresponding messages greatly

depend on a given model's interoperability. Defensive measures against advanced spamming techniques and strategies that solicit the trust of users, and the cybersecurity community must be put into action.

Moreover, previous works have employed static train-test splits, which pose potential model bias and variance problems. This study will adopt k-fold cross-validation to improve the reliability of the model's performance across various test sets. The study assures that the performance metrics do not suffer from dataset bias, which is overly optimistic.

On another topic, Tubishat and Hossein Alipour (2023) explored feature selection of their optimization-based systems but could not apply the more advanced deep learning models that comprehend the internal structure of spam content. This study will fill the gap by employing Keras deep learning models, which are more proficient in the advanced spam patterns and manipulation of spam techniques.

This study seeks to prove the relevance of traditional approaches in the presence of deep learning components and the relevance of such approaches to system performance.

This study attempts to fill the gaps left by the reviewed studies while integrating the highlighted strengths of the prior research. Using feature robustness testing with SHAP-based interpretability, sophisticated validation methods, and deep learning provides a more reliable, explainable, and versatile spam detection system that withstands sophisticated spam attacks.

Specifically, this study targets three gaps: the absence of multiclass feature interdependency evaluation, low explainability of model results, and potential bias resulting from the static training and testing data split. This research aims to build a resilient

and easily understandable spam filter that incorporates logic-driven SHAP explanations, systematic feature removal, k-fold cross-validation, and advanced Keras deep learning models, which will withstand new tactics spammers employ over time.

Chapter 3: Methodology

3.1 Introduction to the Methodological Framework

While developing the spam detection system, this study was not only concerned with optimizing one metric. It takes a broader view – multi-objective optimization is required.

From a methodological point of view, while performance remains important, it is only a part of a broader framework that incorporates interpretability, probabilistic calibration, and sensitivity to key feature changes. This approach stems from the ever-changing spam landscape and email filtering systems' operational requirements. Spam messages can change their tactics at breakneck speed, and a model heavily predicated on narrowband signature attributes set may become useless almost instantaneously once spammers change their plans. In addition, more traditional predictive domains like spam detection need, in addition to good performance, transparent models that account for why particular messages are classified as spam or non-spam. Such models enable administrators to set expectation thresholds confidently and explain why particular messages are classified as spam.

To achieve these goals simultaneously, the methodology in this chapter employs both conventional machine learning (ML) and deep learning (DL) models, providing a comprehensive solution to the task. Random Forest, Gradient Boosting, and a classical Multi-layer Perceptron (MLP) are traditional ML approaches. These methods are appropriate for the Spambase dataset because of its structured numerical features. These methods also offer consistent and interpretable results, such as by feature importance measures. At the same time, deep learning with Keras, as implemented here, is good at

modeling non-linear interactions of the pre-engineered features of the dataset. Such automatic feature extraction helps capture spam features that are very difficult to construct manually. On the other hand, deep learning models, like most other types of models, tend to be overconfident with their probabilities. This framework solves this problem with a post-processing calibration method called temperature scaling, which adjusts predicted probabilities to match the actual outcome frequencies better.

A further fundamental principle of this strategy is the pledge towards interpretability, as achieved through the use of the SHAP (Shapley Additive Explanations) tool. These tools explain and break down a model's output into its constituent parts to show which particular features, such as the frequency of certain words, characters, or even capitalization, disproportionately contributed to detecting spam. The ability to explain and justify the decision is important for model refinement and the trust of the users or the administrators.

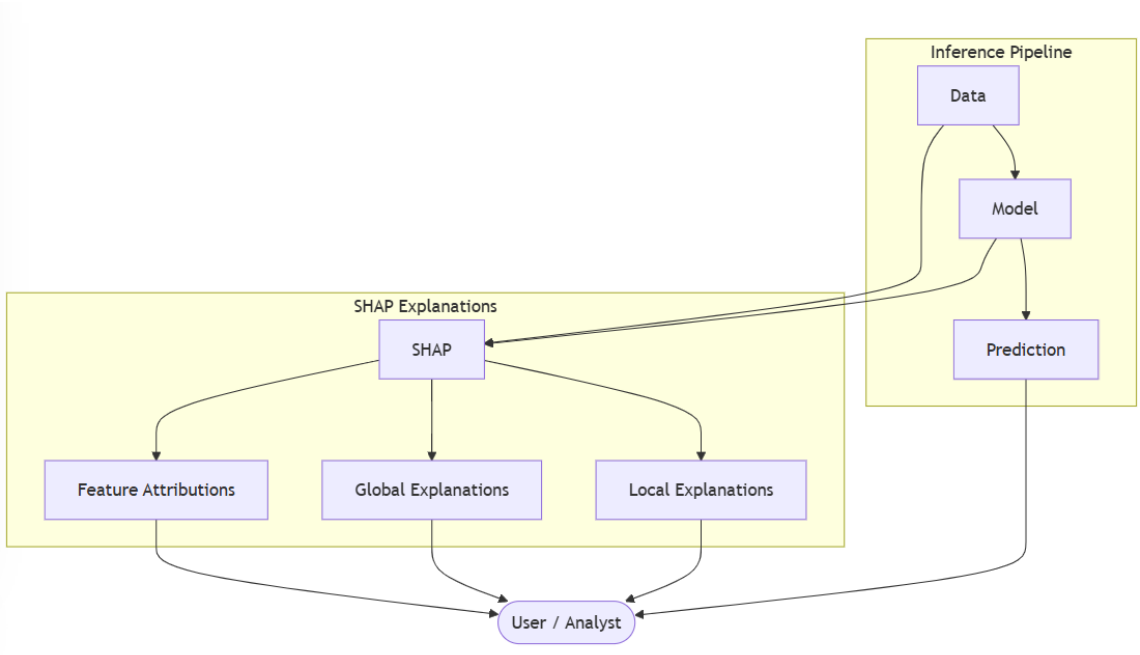


Figure 1: Workflow for Predictions and SHAP Explanations

Furthermore, to guard against overfitting, a so-called resilience test is conducted to check for dependence on one or a few features. This study evaluated how much performance deviance can be incurred before reaching a specified limit. This procedure determines whether learned representations are reliable or depend on a limited set of signals that spammers can easily provide.

Each of these concepts' classification averaged probabilities, reasoning, and robustness—seamlessly integrates into a single methodological pipeline that begins with meticulous data preparation. Even though the Spambase dataset is known and well maintained, it still needs processes such as checking the data's validity, normalizing the data, and exploratory data analysis. After these steps have been completed, each model is progressively trained and then subjected to hyperparameter optimization through stratified k-fold cross-validation to make performance estimates as accurate as possible.

After training, the deep learning model is recalibrated via temperature scaling to make sure that when the model outputs a probability of an email, that probability represents the chance that the email indeed contains spam, ensuring that if the model assigns a certain probability an email, that probability reflects the case of spam in the real world. The last step involves the feature resilience test interpretability analysis, which incorporates SHAP explanations of the models. These tests determine that the system is not operating the decision-making process based on a single feature and can render these decisions in a format non-experts can understand. This holistic approach combines data cleansing, model training, calibration, interpretation, and resilience measures to attempt to fill in the gaps found in the literature.

The delivered spam detection system aims to possess strong classification metrics, probability metric trustworthiness, decision transparency, and adaptability to changes in spam tactics. Every subsequent section will outline each step of the method, describing the interpretation and calibration of models and evaluating resilience and interpretability.

3.2 Data Preparation

The data source that this study centers on is the Spambase corpus, found in the UCI Machine Learning Repository. It is a defined benchmark comprising 4601 emails classified as either spam or non-spam. Each email is represented using 57 numeric features covering textual and stylistic features such as using specific words, counting special characters like "!", "$," "#," and capital letters. These features are crafted to measure elements of language that distinguish spam messages from real ones. Non-spam emails constitute approximately 60%, while spam emails comprise 40%.

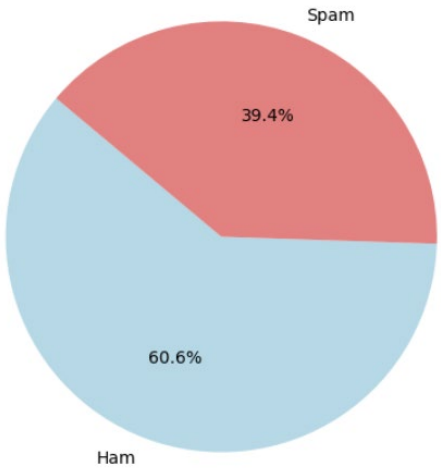


Figure 2: Overall Spam vs. non-spam Distribution

The pie graph above explains the percentage of each class that makes up the Spambase dataset. As can be seen, around 39.4% of the emails are categorized as spam, while the remaining 60.6% are sorted as non-spam or ham emails. Using stratified sampling, the methodology provides 60/40 integer preserving splits for each training and validation fold, which guarantees that the model is always exposed to realistic ratios of spam to ham emails during hyperparameter tuning and performance evaluation. This ensures better estimates of the model's generalization capabilities. Also, the pie chart shows prior data, which proves that even if spam is not the primary category, it does carry considerable weight. Therefore, it supports the case for message classification and validation interpretation.

Spambase has been used many times by researchers looking to test new spam detection techniques due to its structure and numeric feature design, which allows the application of various algorithms, such as decision trees and deep feedforward networks, without the need for text parsing.

Before commencing any analysis, the dataset was systematically scrutinized for any data quality and structure discrepancies. An early scan indicated no gaps or NaN values to be dealt with within the entire dataset of 4,601 records. Each feature's data type was confirmed as correct: most were floating-point numbers (indicating portions or frequencies), whilst some associated with run lengths were integer-valued. This is owing, in part, to how the dataset was collected, which drastically enhanced its cleanliness. Thus, the requirement for more complex cleaning methods like outlier removal or null value replacements was minimal. That said, every feature was looked at on some level through summary statistics calculated from mean, median, standard deviation, range, etc., to

mitigate any extreme values outside the bounds of expectation. Those checks demonstrated that the numeric values were consistent with the known values of spam-labeled emails (for instance, promotional words or special characters had a higher frequency) and non-spam messages (who have less frequent promotional words or special characters).

```
Data columns:
 Index(['word_freq_make', 'word_freq_address', 'word_freq_all', 'word_freq_3d',
       'word_freq_our', 'word_freq_over', 'word_freq_remove',
       'word_freq_internet', 'word_freq_order', 'word_freq_mail',
       'word_freq_receive', 'word_freq_will', 'word_freq_people',
       'word_freq_report', 'word_freq_addresses', 'word_freq_free',
       'word_freq_business', 'word_freq_email', 'word_freq_you',
       'word_freq_credit', 'word_freq_your', 'word_freq_font', 'word_freq_000',
       'word_freq_money', 'word_freq_hp', 'word_freq_hpl', 'word_freq_george',
       'word_freq_650', 'word_freq_lab', 'word_freq_labs', 'word_freq_telnet',
       'word_freq_857', 'word_freq_data', 'word_freq_415', 'word_freq_85',
       'word_freq_technology', 'word_freq_1999', 'word_freq_parts',
       'word_freq_pm', 'word_freq_direct', 'word_freq_cs', 'word_freq_meeting',
       'word_freq_original', 'word_freq_project', 'word_freq_re',
       'word_freq_edu', 'word_freq_table', 'word_freq_conference',
       'char_freq_;', 'char_freq_(', 'char_freq_[', 'char_freq_!',
       'char_freq_$', 'char_freq_#', 'capital_run_length_average',
       'capital_run_length_longest', 'capital_run_length_total', 'spam'],
      dtype='object')
```

Figure 3: Data Columns extracted from the Spambase Dataset

While most machine learning models can work with raw non-scaled numbers, feature scaling was done to improve convergence properties, especially for the heavy learning model built using the Keras deep learning framework. A standard scaling procedure was done by converting every feature x to $(x - \mu) / \sigma$, where μ is the feature's mean, and σ is the standard deviation of the feature. This was done for ease of use in algorithms such as feedforward neural networks, which are more effective with zero-centered inputs and uniform variance. Random Forest and Gradient Boosting models are often called tree models and tend to be less sensitive to unscaled features. Still, using such

a uniform approach with scaling for all models helps to compare results and maintain consistency during further analysis.

An exploratory data analysis was conducted to understand the data features and their possible effects on spam classification. The class distribution confirming about sixty percent for ham and forty percent for spam emphasized the need for stratified sampling to ensure that there is a proportional representation in both the training and test sets. In addition, word_freq_remove and char_freq_$, among other descriptive statistics, ticked spam emails with markedly greater means. This is consistent with the general belief that spam content contains some form of financial or marketing offers.

For detailed definitions of all study variables, please refer to the data dictionary provided in Appendix Section A.1

Correlation analysis explained some feature groups moving together. For example, some capitalization run-length metrics sometimes correlate with one another, meaning that spammers' use of uppercase text often accompanies other forms of stylized writing. While no feature pair was sufficiently correlated to require the elimination of one, these findings contributed to the features' resilience tests, which were devised to assess better the effects of removing or zeroing out attributes with high correlation.

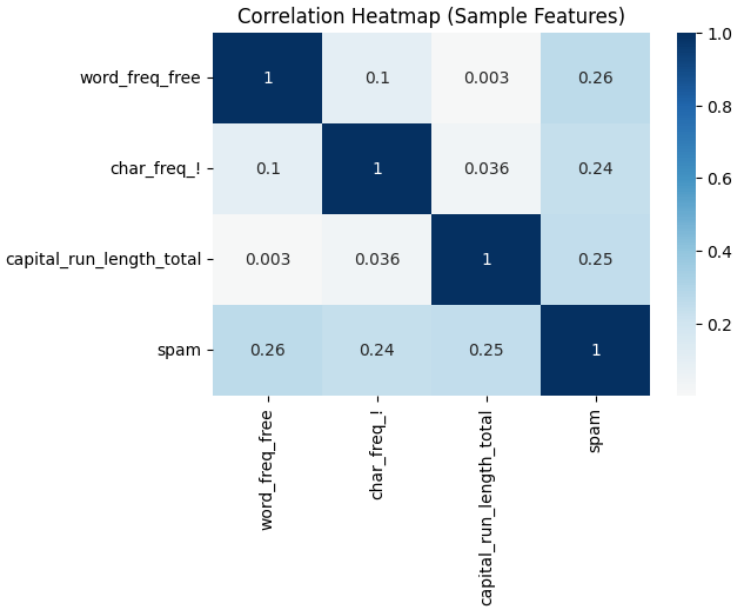


Figure 4: Correlation heatmap for sample features

This sample correlation heatmap above depicts the relationship among a subset of features, i.e., word_freq_free, char_freq_! And capital_run_length_total, and their correlation collectively with the spam label. The color in each cell represents the measure of association to varying degrees. The darker the hue, the stronger the correlation.

Importantly, all three undertake a moderate positive association with spam. Emails having the term free a great deal, along with an abundance of exclamation marks or extensive capitalization runs, are likely to be spam. The relationships among the features say, the correlation between word_freq_free and char_freq_!, suggests that a considerable number of spammers may employ more than one tactic at a time, which is using promotional language combined with striking punctuation or formatting.

Furthermore, these attributes contribute to differing spam detection signals, distinguishing unwanted messages from wanted ones that serve the model. While the

association is not exceptionally massive, the strong positive direction confirms that the features are useful for spam detection. The remaining portions of this chapter and the next chapter will present more details on the feature analysis, the correlation, and feature relevance.

After confirming data integrity and applying optional scaling, the data was split into training and test sets with an 80:20 ratio. Stratified sampling was used to maintain the class proportion so that both subsets preserved the approximate 60:40 ham-to-spam ratio. This technique is important in spam detection because performance can drastically differ with unrepresentative class proportions within the training or test set. The training set was then set aside for model development involving hyperparameter optimization and cross-validation. The test set was exclusively allocated for final evaluation. This approach is under best practices as it minimizes data leakage and provides inflated estimates due to using the exact data for tuning and final testing.

The Spambase dataset's structure and its numeric features allowed for the complete range of experiments within this study to be conducted seamlessly. Its consistency and absence of null values damaged the less elaborate data-cleaning pipelines' creativity, which was further shifted towards more sophisticated structural components like model tuning, interpretability, and robustness. This study established a strong base for evaluating classical and deep learning approaches for spam detection by performing careful data validation, standard scaling for uniformity, and preserving a reasonable class ratio in training and test sets. The following sections elaborate on the specifics of model choice, training plans, and custom strategies, like temperature scaling and feature resilience tests, that constitute the backbone of this methodological framework.

3.3 Model Selection and Justification

3.3.1 Traditional Machine Learning Models

The choices made on model selection for spam detection are more complex than choosing a model that has performed well in the past. It is also about interpretability and transforming the model features into numeric values. Regarding the Spambase dataset, which has 57 constructed numeric features, traditional machine learning models are a great starting point. These models have a strong history of performing reasonably well on structured data, with robust spam detection capabilities and explanatory or visual mechanisms of feature importance. This research selected three traditional ML models: Random Forest, Gradient Boosting, and a classical Multilayer Perceptron (MLP). Each has its strengths and weaknesses relative to the other concerning spam detection, especially considering the structured nature of Spambase's word frequencies, character frequencies, and capitalization ratios.

Not only is Random Forest recognized for constructing numerous decision trees, but it also trains each one on random subsets of both data and features. This ensemble method reduces the overwhelming overfitting that single trees tend to have. The model is now more stable and able to withstand noise or redundant features, which is typical in email data. Certain words or character frequencies overlap and correlate. Random Forest smoothens noise and lowers the variance of the effect caused by any of the individual trees by averaging the outputs of multiple trees. When dealing with spam detection, one tree may overemphasize word_freq_free, but another tree can counterbalance that bias by utilizing char_freq_! or capital_run_length_total. Moreover, most of the time, Random

Forest is known to output feature importance scores. This allows model users to comprehend which attributes word_freq_remove and amped char_freq_$ are most powerful in driving the classification. The ability to comprehend the model's inner workings is functional, especially when diagnosing model errors or changing the filters to counter emerging spam tactics.

Like Random Forest, Gradient Boosting combines numerous decision trees but uses a different approach for assembling them. Rather than training all trees in parallel on random subsets of data, Gradient Boosting modifies a "weak learner" with each iteration, emphasizing the previously made errors. Greater performance is achieved each time the ensemble makes fewer " misclassifications, " although it requires more training time. For spam detection, this focus could be tremendously helpful in cases involving intricate patterns, such as particular punctuation or capitalization, that tend to stump earlier trees.

In these cases, Gradient Boosting can become adept at observing subtler rules that more uniform methods would miss. In practice, hyperparameters like learning rate, boosting rounds, number of trees, and maximum tree depth require extreme care in tuning, lest they promote overfitting, especially when dealing with correlated features in a dataset like Spambase. Despite this challenge, and after being correctly tuned, Gradient Boosting performs exceptionally well in most classification tasks with tabular data and provides feature importance metrics for easier interpretation.

The classical MLP in this study is situated somewhere between tree-based ensembles and deep learning. Its deployment is well-known in the scikit-learn libraries, and it achieves non-linear decision boundaries owing to several layers of perceptions with activation functions like sigmoid or ReLU. Unlike tree-ensembles, MLPs usually scale

better with feature redundancy, and if deep enough with regularization, they can capture intricate interdependencies among the variables. Still, some of the more sophisticated features and personalization options available in the Keras, such as frameworks, are unavailable. However, it is helpful to add a classical MLP because it provides a fundamental perspective of a neural network that complements the ensemble approaches.

In particular, it helps to understand if the supposed improved performance of state-of-the-art deep learning models actually comes from having a more complex architecture and elaborate training processes or if simpler neural networks with the numeric variables in Spambase achieve a sufficient level of classification success.

The study aims to incorporate a wide range of approaches for processing numeric information by selecting these three models. As their name suggests, Random Forest and Gradient Boosting are well known for being restrained towards unscaled features and delivering rather good results on datasets with average dimensions. Classical MLP neural networks, for instance, depend on feature normalization and can capture non-linear relationships by adjusting specific network parameters. All three models provided are allowed for fair evaluation using the same train/test splits, similar cross-validation designs, and identical performance measures (accuracy, F1 score, ROC AUC), increasing the comparison's reliability. Moreover, the models can be used in the other stages of the methodology, which are the calibration of probabilities and testing the resilience of features. With these additional layers of analysis, the research attempts to answer how accurately the models classify spam and non-spam emails, their predicted probabilities, and how highly the features depend on them.

One more topic related to the adoption of these models is their interpretability. Although none is self-explanatory, both Random Forest and Gradient Boosting generate feature importance estimates, and the classical MLP can be partially understood via sensitivity analysis or partial dependence plots. That sort of understanding becomes critical in adversarial domains such as spam detection, where it is important to know how a model makes a classification to improve filters and predict evasion attempts. For example, if the model over-relies on a single feature (char_freq_!), spammers may decide to remove or obfuscate that character. Understanding which features are most important helps security teams mitigate risks by varying their filtering methods ahead of time.

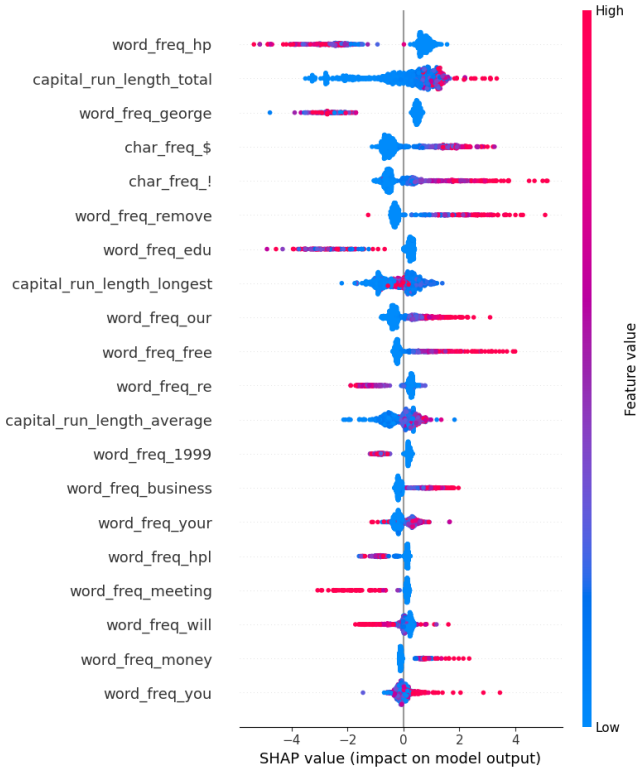


Figure 5: SHAP Summary Plot for Gradient Boosting

For instance, the SHAP summary dot plot above shows how each feature contributes to the spam classification by the Gradient Boosting model at the most granular level. Each dot represents one email and is located on the x-axis according to its SHAP value, which is how much that feature either pulls the prediction toward spam (positive SHAP) or pushes it away from spam (negative SHAP). The dot color indicates the feature's untransformed value for that email, ranging from low (blue) to high (pink or red). Thus, features that tend to produce high SHAP values on the positive side almost always correspond to strong spam classification. In contrast, features with negative or lower SHAP values correspond to ham or a weak effect.

Primary attention is drawn to word_freq_hp and capital_run_length_total, which appear to possess many positive SHAP values. This suggests that these features are important in increasing the model's estimation of spam probability. This supports the hypothesis that some specific words (like "hp") and uppercase letters spam marker words are predictive of spam messages. In contrast, features such as word_freq_you or word_freq_money are more centralized, which means their impact is much weaker or contextually dependent. Moreover, the color gradient within a band shows how the feature's raw values are associated with changes in the model's output; for example, high values of "hp" (depicted with pink or red dots) tend to result in strongly positive SHAP values while low values (blue) usually results in less positive or negative values. This summary plot allows one to see the tradeoff between position and color, which captures the extent and polarities of each feature's contribution to individual predictions. This provides a better understanding of the model's reasoning throughout the dataset.

To conclude, Random Forest, Gradient Boosting, and a classical MLP constitute the traditional ML framework of the approach taken in this study. Their selection was based on prior success with numeric datasets, the ease of interpreting results (particularly with ensemble trees), and differing degrees of highly intricate feature interactions. The purpose of these models is to measure performance with a single calibration and validation framework, which makes their performance and estimate reliability alongside the robustness of the learned feature representations and probe subjects. The feature and feature representation learning are tuned to operate in spam detection with utmost extremity with a Keras-based neural network, and the network layers provide the least depth. This allows for comparison using deep learning approaches, which are centered around the comparison of the neural and the ensemble tree.

3.3.2 Keras-Based Deep Learning

Although classic machine learning approaches are practical with tabular data because they are easily interpretable, deep learning methods can capture interactions among features in a dataset more complexly and nonlinearly. This is beneficial for datasets such as Spambase. For deep learning development in this study, Keras was chosen because of its flexibility, ease of use, and ability to connect seamlessly with the other Python data ecosystem tools. Keras allows users to use a high-level API that simplifies the implementation of the conceptual model architecture, allowing researchers to focus on important parts of the experiment instead of micromanaging implementation specifics. In the case of spam detection, where the textual patterns to identify can be subtle, overlapping, or even adversarial, the ability to rapidly prototype several network configurations is

essential. These configurations may include the number of hidden layers, the type of activation function used, and the dropout rate.

The Keras framework has the inherent advantage of building feedforward neural networks (MLPs or multilayer perceptions) that deal with structured numeric data. Even though deep learning is primarily concerned with images and raw text, MLP architectures, whose structures and topologies are considered deep, can still accommodate the numeric features of Spambase. Each email in the dataset is represented through a set of features, which includes the word_freq_remove, word_freq_free, and char_freq_! Frequencies along with the capital_run_length_total. The underlying complexity within and across multiple hidden layers can be learned by passing these numeric features using the neural network. For instance, specific deeper layers of the network may use holistic interaction patterns between some allowed-outside-punctuation and run-length capitalization mixed with the overall view of spam indicators to enable neurons potentially.

The Kears framework also makes implementing advanced training strategies that improve the model's generalization and overfitting capabilities easier. For example, batch normalization can be included between layers to improve convergence rates by stabilizing the distribution of activations. Dropout enables the random deactivation of a certain percentage of neurons during training, which helps to promote the network's robustness by developing redundant internal representations. In the case of spam detection, where spammers constantly change their strategies, this regularization may result in a more robust network to feature shifts or noise.

Besides these features for the trainees, Keras helps with probability calibration and interpretability. Specifically, the library's modular architecture allows temperature scaling

and other post-processing calibration procedures to be applied to the raw output logits from the network's last layer without contamination. This feature enables the study to treat calibration as an supplementary task, adjusting the model's probability outputs without changing the classification boundary. Since deep neural networks are known to be overconfident in their predictions, calibration will increase the practical usefulness of these systems. This is especially true when system administrators wait for some probability value to decide whether to block some emails or quarantine them.

Keras-based models can be integrated with SHAP, which explains how each feature affects a given prediction. While feedforward networks are inherently less interpretable than decision trees, the combination of Keras with XAI tools answers the problem of the 'black box.' With the help of SHAP, the outputs can be represented in terms of numeric signals (like char_freq_$ or word_freq_000), which classify the output as the strongest.

Such knowledge is instrumental in feature engineering, and help ensuring that the system logic is preserved with the known facts about the content classified as spam.

Another advantage of Keras is its effortless integration with GPU acceleration and the use of techniques for distributed computing. This benefit might not be very important for medium-sized datasets such as Spambase, but it allows for scaling up if required.

Keras can detect more sophisticated embeddings or even hybrid CNN-LSTM architecture changes if one switches to raw text inputs, as spam detection works, to expand and larger corpora become available, potentially containing raw textual or multi-lingual data. Even within the numeric domain, multi-GPU support facilitates faster

experimentation, which allows for more comprehensive hyperparameter searches that would be difficult to conduct using only a CPU.

Ultimately, Keras is a valuable addition to this project as it enables the integration with classical ML. Keras models with tree-based algorithms using blending or stacking can easily be incorporated. This possibility supports the broader aim of the research, which is to investigate how different approaches can be consolidated for spam detection by using ensemble trees' interpretability and deep learning's representational power. The decision to use Keras as a deep learning framework serves to provide the study with a contemporary and flexible platform that goes beyond the immediate scope of the research, which centers on strong performance, calibration, and interpretability, but also allows for future analysis concerning more advanced or data-rich scenarios of spam detection.

3.4 Hybrid Considerations

Even though this study analyzes a standard set of machine learning model implementations alongside a Keras deep learning model, a growing body of literature argues that the most optimal spam detection solutions are likely to come from integrating multiple models into one cohesive system. This hybridization can take many shapes, from basic voting ensembles to more complex “stacking” systems, wherein one or several “meta-classifiers” are trained to optimally combine base models' predictions. There is strong reasoning in trying to apply these approaches to spam detection: for the most part, tree-based ensemble methods are very good at capturing and allowing the interpretation of strong patterns in numeric data, and deep neural networks are good at capturing intricate, non-linear interactions which are usually missed by the more rigid algorithms.

One option for combining approaches is to leverage the different ways each model extracts features for its own benefit. For instance, a Random Forest may repeatedly emphasize specific word frequency measures as primary correlates of a text. However, a deep network built on Keras could detect intricate relationships between run-length totals of capitalized words and specific punctuation mark frequencies. Some hybrid systems have proven capable of correcting some known effects for single algorithms, like the overfitting of neural networks or the underfitting in simple ensemble systems, by having each constituent part contribute to a meta-classifier that determines the optimal decision boundary.

Another motivation for hybrid systems is that they can enable more refined interpretability. While deep learning architecture can be opaque, it can be integrated with more interpretable models (like Gradient Boosting or Random Forests). For instance, if both the deep network and the Random Forest always place word_freq_remove amongst the top signals, it strengthens the case that this feature is highly predictive across models.

On the other hand, if the deep network captures a strong interaction while the Random Forest appears to miss it, it might motivate feature engineering or some form of analysis to see if the explicit inclusion of that interaction can improve the ensemble.

Nonetheless, developing a hybrid model poses both practical and theoretical obstacles. It may involve some level of orchestration of data ecosystems since each model might require specific algorithmic preprocessing (e.g., scaling features for neural networks and not for tree-based models) or particular hyperparameter tuning methods.

Furthermore, the final ensemble must be validated by cross-validation methods representing both model outputs unbiasedly. This is particularly important in areas such as spam detection where moderate class imbalance (approximately 60% ham, 40% spam) can adversely affect performance evaluations if not appropriately managed.

Even with the challenges posed by integrating a hybrid system into spam detection, the resultant advantages could be considerable, particularly in the context of the meticulously crafted Spambase dataset. An optimal ensemble might capture the benefits derived from representing a Keras-based neural network using Gradient Boosting, thus achieving good classification results and well-calibrated substantial probabilities. Such a system could also support feature resilience checks so that the tree-based part and the neural network do not dominate the system by being too dependent on a single feature. If the combined framework provides additional interpretability via SHAP or partial dependence plots, it could explain the contribution of each model toward the outcome itself, which can lead to an understanding of spam content and the classification process.

3.5 Model Training and Hyperparameter Tuning

3.5.1 Tuning Parameters for Traditional ML

How models are trained effectively mostly depends on defining hyperparameters for algorithms that dictate how each method assimilates information. Regarding traditional machine learning methodologies like Random Forest, Gradient Boosting, and a classical Multilayer Perceptron (MLP), this process can enhance predictive performance in addition to the overall interpretive value of the model. While every algorithm has specific hyperparameters, they all share the common objective of achieving a compromise between

meticulously fitting the training data and being able to generalize on unseen emails. Since the Spambase dataset has 57 numeric attributes representing different textual features, it is imperative to manage these hyperparameters so that each model does not overfit the dataset's intricacies or specific features and spam patterns.

In the case of Random Forest, the most important considerations for hyperparameter tuning are the number of decision trees, the allowed maximum depth of each tree, and the node splitting criterion. The number of trees is typically referred to as n_estimators. The stability of a model is usually improved with an increase in the number of trees because more decision paths are averaged; however, training times will also increase. The maximum depth, max_depth, is how deep each tree goes in partitioning the feature space; if trees are too deep, there is a risk of overfitting due to the tree being too deeply entrenched in the peculiarities of the training set, while trees can underfit if the data possesses complex interactions. Other key hyperparameters include min_samples_split, which is the minimum sample number required in order to perform a split on an internal node, and min_samples_leaf, which is the minimum number of samples required in order to be at a leaf node. These parameters control the size of the tree and help ensure that partitions are not made on very small subsets of data, which in turn helps prevent inflating variance. Additional implementations permit the tuning of max_features. For Spambase, which has 57 features, tuning max_features will help reduce redundancy and allow each tree to try to use other subsets of attributes, thus improving ensemble diversity.

The tuning of hyperparameters is revisited several times in Gradient Boosting as it has more of an iterative logic. In contrast to Random Forest, which trains each tree independently at the same time using bootstrapped samples, Gradient Boosting adds trees

one at a time and aims to correct the residual errors left unaddressed by the previous models. Hyperparameters include the contribution of a single tree complicated by a newly added tree scaled by the learning rate and the number of boosting stages n_estimators. Each incremental update is more conservative, so a smaller learning rate usually needs more boosting rounds to reach convergence, although it is often more effective when it comes to generalization. Moreover, max_depth assists in determining how each weak learner divides the data over an extended limit. If there is too much depth, trees can pick up very specific spam patterns, such as particular punctuation with a combination of its frequency and overfitting these patterns, not generalizing. Due to the intricacy of the boosting process, these hyperparameters frequently have relations with one other; for example, if there is a reduction in the subsample, it can increase n_estimators to try and improve performance.

The Keras-based implementation of MLP scrub out some hyperparameter considerations from the classical approach to the reconstruction process. Single MLP Hyperparameters include the acute number of hidden layers, the number of neurons on each layer, the activation function needed like ReLU, sigmoid, and regularization method like L2 weight decay and dropout. Hidden layers in the provided range might suffice to capture a moderate number of interactions among features due to a moderately sized and entirely numeric Spambase dataset without overly complicating the architecture.

In tune-up hyperparameters for most of these models, traditional approaches utilize a combination of systematic search. A popular method is grid search, which entails a set of parameter limits that must be tested exhaustively. For example, to evaluate a Random Forest, assume n_estimators need to be set in the set {100, 200, 300} and max_depth in the set {5, 10, 15} for all possible combinations using cross-validation.

While thorough, this method becomes computationally expensive with an increased number of parameters. Alternatively, the random search can stochastically select configurations for a parameter in a given range while more efficiently covering regions of the hyperparameter space.

Given that the Spambase dataset is moderately sized (~4,601 samples), these search strategies are manageable computationally, and repeated cross-validation can compound in runtime. During this study, stratified k-fold cross-validation is conducted to cover all folds on which each model configuration is tested, preserving the ~60/40 class ratio. This practice instills reliable estimates of performance, such as the mean F1-score or mean ROC-AUC, and provides measures of how reliable each hyperparameter setting is across different subsets of data.

After assessing all models and selecting an optimal or near-optimal configuration for each, the final retraining happens on the full training set. It is followed by scoring on the held-out test set. Both Random Forest and Gradient Boosting produce calculable feature importance scores, and hyperparameter settings that produce a feature importance score distribution are balanced enough to minimize over-dependence on a single feature.

In the same way, with the classical MLP, weight magnitudes and partial dependence analyses permit checking whether hyperparameter settings make the network get stuck on some dominant high-frequency features like char_freq_! While missing other features that are predictive.

The tuning of hyperparameters for the Random Forest, Gradient Boosting, and classical MLP is done in steps to get the most efficient output from each algorithm with

appropriate reasoning and no overfitting. By changing values of different variables like the number of trees and learning rates, hidden layer sizes, etc., and guided by the cross-validation metrics, this study makes sure that the final configuration for each model is appropriate for narrow numeric partial correlation and a moderate-level of imbalance of the Spambase dataset. Such a rigid framework for a more integrated methodological approach helps in probability calibration and feature resilience testing and, ultimately, in expanding the limits these traditional models can work with, especially in complex tasks like spam detection.

3.5.2 Keras Deep Learning Frameworks data processing

Constructing a deep learning framework for spam detection requires finding a delicate equilibrium between the model's sophistication and the structured, numerical Spambase dataset. Deep neural networks are usually reserved for unstructured data types, such as pictures or sequences of text. However, they can be effectively trained on tabular features with adequate configuration and tuning. Keras, in this case, is the selected framework for deep learning due to its high-level API, which allows for quick prototyping and full integration of important features, including dropout, batch normalization, and sophisticated optimization algorithms.

For Keras implementation, the architecture begins with a feedforward network (Multilayer Perceptron, MLP) that takes in the 57 numeric features from the Spambase dataset. Each email is represented as the frequency of chosen words (e.g., word_freq_remove, word_freq_free), the frequency of characters (e.g., char_freq_!), and run-length statistics (e.g., capital_run_length_total). It is designed to construct an input

layer that accepts a vector with 57 components to incorporate these inputs optimally. This layer is followed by one or multiple hidden layers containing neurons that perform mathematical operations on the data and output the result through an activation function, such as ReLU (Rectified Linear Unit) or Sigmoid. ReLU is greatly used for numeric data because it is simple and does not suffer from the vanishing gradient problem.

The depth and width of the network are important because the primary design will ultimately hinge upon how many hidden layers there are and how many neurons are assigned to each layer. Spambase's samples require careful neural network regularization to avoid overfitting, which would be very difficult with less than 5,000 samples. More complicated deep networks might perform better in capturing non-linear feature interactions, but their architecture requires more intricate regulating. The architecture usually uses 2–3 hidden 32–128 neuron layers. The bounds on layer widths are often defined by some form of heuristic rationale coupled with systematic hyperparameter tuning or solely by reasoning heuristics. As the output layer, a single neuron scaled with a sigmoid function is used to indicate the probability of a spam email. Overfitting can be tackled by employing various regularization methods within Keras.

Dropout decreases dependence on individual neurons by randomly zeroing a fraction of them in each training batch, which makes the network train better-generalized representations. For example, a 20–30% dropout rate on hidden layers means that 0.2 - 0.5 of neurons are zeroed out in each training batch.

On the other hand, batch normalization adjusts each layer's output by removing the batch mean and dividing it by batch standard deviation, in addition to applying learnable shift and scale parameters afterward. In real-world applications, batch normalization can

stabilize the distribution of activations, making training faster and, in some cases, improving performance. For numeric data like Spambase, it helps maintain feature scale in check, thus aiding any prior normalization done at the pre-processing stage.

The other method is L2 Regularization, which is also called weight decay. It penalizes the sum of squared weights for L2 regularization, which encourages the model to keep its parameters small. This is useful when the network tends to create large weights for some spam indicators, such as too much emphasis on certain punctuation frequency, while ignoring many other important features.

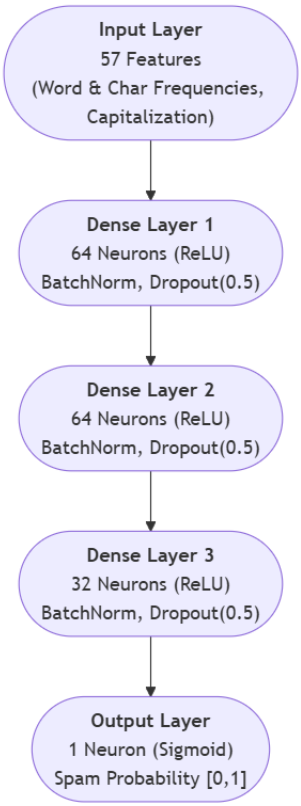


Figure 6: Neural Network Architecture for Spam Classification

Feeding this network requires optimizing to change the model's parameters based on the loss. Among other optimizers, Adam is often selected since it combines the advantages of both AdaGrad and RMSProp. It modifies the learning rates for every parameter individually based on the first and second moments of the gradients. Adam's default values of hyperparameters ($\beta_1 = 0.9$, $\beta_2 = 0.99$, and $\varepsilon = 10^{-8}$) are okay for most cases, although the initial learning rate almost always needs adjustment. Inadequate tuning can result in overly aggressive, unstable, or sub-converging training with prolonged tuned low learning rates.

To maximize spam characteristics, one would generally start with a learning rate of 1e-3 to 1e-4, but this has to be modified through cross-validation (Diederik et al., 2014).

3.6 Calibration and Explainability

An issue that needs to be tackled with deep neural networks is that they frequently output highly uncalibrated probability estimates, which signifies that the predicted probability does not match the actual spam probability. This flaw is fixed further down the pipeline in temperature scaling, where a single scalar parameter controls logits.

Analyzing Shapley values for a given set of features explains how specific numeric signals, such as word_freq_remove or capital_run_length_total, impact the network's final computation of spam probability. This combination of sophisticated neural techniques and interpretative tools enhances reliance and gives clear guidance on improving the feature set.

Subsequently, the precise layer dimensions, dropout percentages, and optimizer settings are adjusted through cross-validation. After the model has converged, it is retrained on the entire training set using the best hyperparameters. The validation metrics are

calculated with the held-out test set, where accuracy, F1 score, and ROC AUC are all evaluated. After this step, the pipeline moves on to calibration (temperature scaling) and stress testing with the contended deep network to check whether the assumption is valid confidence score calibration and the reliance on a feature.

The presented methodology uses Keras to fully utilize the neural network's functionality for numeric spam characteristics, while overconfidence and opacity risks are handled via calibration and interpretability tools. Striking a balance between neural network sophistication and strong downstream processing sufficiency defines the essence of building a trustworthy and adaptable spam detection system for spoofing.

3.7 Cross-Validation and Evaluation Metrics

After the hyperparameters of both traditional ML and deep learning models are set, the methodology relies on a comprehensive evaluation protocol that reports the degree of generalization and performance for spam detection for each model. An important component of this protocol is stratified k-fold cross-validation. This technique splits the training set into k approximately equal parts (folds) while maintaining the ratio of non-spam to spam emails. Each fold serves as a temporary validation set. In contrast, the remaining k−1 folds constitute the training subset, which permits the model to be trained and evaluated several times on different parts of the data. The study enables more accurate estimates of how well the model performs with previously unseen examples by averaging performance metrics (accuracy, F1-score, ROC AUC score) over all folds. This allows for minimizing the possibility that any single data partition might misrepresent the model's performance, which is particularly important in spam detection, where moderate class

imbalance can result in overly optimistic measures if not carefully controlled during both training and validation phases.

The F1-score remains fundamental within this analysis, representing the harmonic mean of precision (the ratio of correctly predicted spam emails to the total number of emails predicted as spam, i.e., true positives plus false positives) and recall (the ratio of correctly identified spam emails to the total number of actual spam emails, i.e., true positives plus false negatives). In spam detection, the consequences of both positive and negative errors are costly due to the potential of legitimate emails being blocked, resulting in productivity loss and spam emails getting through, posing a security threat.

Therefore, the F1 score is a more helpful metric as it neutralizes the advantage of an outperformance in either direction. Accuracy can also serve as a basic measure of the total number of emails correctly classified; however, it has its drawbacks with class imbalance, as a model can achieve high predicted accuracy by forecasting only the majority class. As with any class imbalance measure, ROC-AUC qualifies how well the model distinguishes between spam and non-spam for any given threshold and provides a different view of the model's ability to rank. This is particularly useful when the administrators or researchers want to adjust the probability thresholds for stopping or isolating emails.

Random Forest, Gradient Boosting, classical MLP, and Keras-based deep network models are all trained k times (where k equals the number of folds), which results in many different performance metrics estimates. This guarantees that each sample from the training set acts as a validation sample precisely once, effectively reducing the variance of the performance scores. The folds themselves are constructed through stratified sampling so that the approximate 60 to 40 hostile mails to spam mail ratio is maintained in each fold to

reflect the original distribution in the dataset. After cross-validation, each model was tested with the best-performing hyperparameter configuration and retrained using the whole training set, which was combined for each k fold. This maximally exposes the model to training data, improving the predictive power of the final model models. After this step, a pre-defined held-out test set, which was not used during training or validation folds, tests the model's performance. In turn, this guarantees that the final reported measures—accuracy, F1 score, and ROC AUC score- are overly confident results based on repeatedly exposing them to the same data. In addition to these core metrics, confusion matrices are constructed to assist in interpreting the model's true positives, true negatives, false positives, and false negatives.

This dissection shows whether a model is more likely to block valid emails (false positives) or allow spam emails to get through (false negatives). At times, the study focuses on the "high confidence" region, which is why heaped or ignored messages are likely to be discarded, leading to the examination of precision-recall curves and AUC (average precision). Also, these curves plot probabilities against spam likelihood because calibration techniques, outlined in the next section, check if the model's predicted probabilities align with the actual spam probabilities. With these metrics and visualization aids, the methodology constructs a multi-faceted approach to assessing how each model deals with the numeric attributes of Spambase and the confidence level of the model in classifying spam and non-spam emails. This posture illustrates that the achievement of detection of SPAM emails is not only about attaining the highest measure of merit but one that balances email filtering constraints, which translate to cost-sensitive misclassifications and lack of required interpretation.

3.8 Probability Calibration

Spam detection in practice is often much more involved than simply looking at classification accuracy, F1 score, or even the more advanced metrics. In addition to detailing non-spam versus spam emails, administrators typically use an email's spam probability to make threshold-based decisions. To illustrate, a mail server may automatically delete emails for which the server estimates the probability of receiving spam is greater than 0.90 and place emails with a likelihood of non-spam between 0.70 and 0.90 into quarantine for further evaluation. In order to construct a decision-making process, a model's probability estimates need to be calibrated. Thus, if the model estimates that it has a spam email with a probability of 70%, then approximately 70% of those emails should actually be spam. Such models improve the selection of thresholds, enhance trust in confidence scores, and minimize the risk of mishandling emails when the decision to classify is made due to over or underestimating the model's confidence.

Even though calibration is fundamental, it is most frequently ignored, especially in the case of deep learning models, which have been shown to have well-separated decision boundaries but modest probabilistic calibration. This gap can exist due to the behavior of neural networks, particularly those trained with a standard cross-entropy loss function, which is among the frequently utilized tasks in multi-class classification problems, especially in training deep neural networks. Its focus and ease in training tend to drive the outputs to extreme values (for example, to 0.0 or 1.0) to cut down on classification mistakes (Demirkaya et al., 2020). While these extreme levels of confidence may be beneficial in

maximizing the performance of a training set, they misrepresent the actual probability of being spam. To counteract this, the proposed methodology in this research developed a calibration procedure during post-processing using a deep learning model based on Keras. However, it can be implemented with any classifier that produces probability estimates.

This procedure improves the model's ability to be practical in threshold- and risk-based scenarios by ensuring that the estimated probabilities of events correspond to the true frequencies of the outcomes.

The central concept used here involves temperature scaling, a powerful post-processing technique. This method utilizes a scalar parameter, called T, to divide the logits into a binary classifier. Temperature scaling is an enhancement of Platt scaling by a de-calibration factor T > zero, which controls the logits before applying the SoftMax function. For a logit vector, the new confidence prediction $\hat{Q}_i$ is:

$$\hat{Q}_i = \max_k \sigma SM(\frac{z_i}{T})(k)$$

Where σSM is a SoftMax operator, if T=1 the distribution stays the same. Values of T > 1 "soften" the probabilities, that is, increase entropy, while T < 1 makes them "sharper." Most importantly, softening or sharpening the logits does not change the predicted class, which still has the highest probability. Therefore, model performance does not change, and the estimate is improved to be more realistic (Guo et al., 2017).

When T equals 1, the probabilities are consistent and reflect the model's biases, which are overconfident predictions. In the event that the deep learning model is perpetually overconfident, an ideal T greater than 1 exists, which can "soften" the logits and thus make them closer to realistic outcome frequencies. On the other hand, if the model

is underconfident, which is less common but not impossible, then there is a chance that T less than one could work as it might push probabilities towards higher than realistic values.

The ideal temperature T is calculated by using some form of minimization through a calibration loss, which can be a negative log-likelihood on a calibration set. With this in mind, in practice, what is done is to take the network's predicted logits for a batch of emails and search for the T that best aligns the logits with the true spam labels. The final calibrated model maintains its original classification decisions or threshold-based labels; however, it now reports the probability that more accurately estimates the likelihood of spam.

Simplicity is one of the advantages of temperature scaling. It is more advanced than other calibration techniques, such as Platt scaling or isotonic regression, as temperature scaling adjusts the network’s confidence. This method further does not change the decision boundary or need additional parameters. This single-parameter approach is desirable for deep neural networks trained on moderately sized datasets like Spambase since it does not require overfitting to a small calibration set. It is also advantageous because it only requires the raw logits from the final layer of the network, which means it can be done after the fact without needing to retrain the model. In the workflow of this study, the deep learning model is initially trained in its entirety with cross-validation to find the best hyperparameters and epochs, and only after those are considered is the temperature parameter modified and optimized.

Emails above the 0.50 threshold before calibration stay within the spam classification if their recalibrated probability still exceeds 0.50. However, some argue that the precise numeric probability may change. This is advantageous for practical spam detection systems because it enables administrators to keep the same classification limits

while having a more accurate confidence estimate. Suppose an email is assigned a calibrated probability of 0.85. In that case, it can now be interpreted as an 85% chance of actually being spam, unlike what the system would have thought before calibration, which was 0.95 or 0.99, which would not have been the case.

After calibration, the next step in the methodology is to assess enhancements using calibration curves and tables. A calibration curve shows the predicted probability (x-axis) of a given event versus the observed frequency of its occurrence (y-axis). A model is considered well calibrated if it aligns with the diagonal line of y = x, which denotes that a predicted probability of p matches p-associated spam email prediction.

Observing how the curve changes from “before calibration” to “after calibration” indicates whether the network was overconfident or underconfident at specific probabilities (like 0.7–0.8) and how much temperature scaling helps correct that mismatch. In the same way, a calibration table places emails in different bins defined by ranges of predicted probabilities (0.0 to 0.1, 0.1 to 0.2 … 0.9 to 1.0); for every bin, a mean predicted probability, and the fraction of positives is defined. The difference between these two values (often called the “gap”) is a quantitative measure of calibration error.

The methodology employed here uses calibration of probability estimates through temperature scaling that forms a part of the solution for the frequent problem of overly confident deep neural networks, which is their structural characteristic. It also improves the utility of the Keras model in spam contexts by ensuring that the predicted probability reflects real-world frequencies of spam as closely as possible. The model helps administrators by providing output probabilities they can use to set thresholds or make risk-

sensitive decisions, safe in the knowledge that the model's confidence scores have been adjusted to reality based on the actual empirical data.

3.9 Feature Resilience Testing

When it comes to spam detection, the performance of a model to endure specific changes – removal or alteration of critical features – is critical, especially within a context where spammers are endlessly changing their tactics. Feature resilience testing looks into the specific and very precise consequences of every model's attribute, like word frequencies or punctuation usage, to the overall classification performance of the model.

The reasoning behind this approach is that a spam filter with stable functionality should be generically robust and not rely on a single measly signal. For example, if a model highly depends on char_freq_! Any change to the punctuation style of spam emails may render the filter ineffective. Feature resilience testing, in contrast, seeks to demonstrate the model's potential weaknesses and the range of signals upon which generals depend.

In this study, the tests of feature resilience have a preparatory step, which includes cutting off some features of the input data and modeling the F1-score (or any other metrics). In this case, the tests were performed on two groups of features:

1. The model tests high-valued features the most, as captured through feature importance (for models like Random Forest or Gradient Boosting) or interpretative evaluations like SHAP for deep learning models.
2. Arbitrarily chosen features for use as a control group in an experiment.

The difference in performance drops measured in these two cases would reveal whether the model relies disproportionately on high-ranking features or has learned a more sparsely populated representation. To illustrate, if the test reveals that the removal of word_freq_remove leads to a drastic drop in F1-score for the Keras-based model but a small drop for Random Forest, that difference is an indicator that the deep network depends more on that signal than the ensemble model.

Let us assume $F1_{Original}\ is$ the performance of the model with all features enabled. If one or more features are set to zero and re-evaluate the performance, that would be $F1_{Modified}$. The performance drop is mathematically represented as:

$$\Delta F1 = F1_{Original} - F1_{Modified}$$

If a small interval of time is analyzed where a certain measurement is taken and found to be negative, then removing that feature does not fail the model's spam classification ability, which suggests resilience. On the other hand, large measures of delta indicate high dependence on the feature, which signals problems as to how the model will perform on manipulations of spam signals that fail or mask the feature.

Examining resilience tests in context can be helpful for both modelers and security personnel because it suggests how these groups might align their efforts. A model that is considered robust, in the sense that it has high F1 scores with none of its top-ranked features, suggests that there is useful knowledge of spam cues that is widely available. Such a model would continue to work until spammers figure out how to modify certain lexical or stylistic features in order to be undetected. However, the model suffers sharply in performance from removing a single feature. In that case, the model may have to be

changed further by, for instance, adding more alternative signals or enhancing the diversity of the training data. Further, knowing which features are essential as compared to irrelevant enables the feature engineering process as it allows the study to ease or complicate the feature set and focus towards better the feature set's responsiveness.

Reconciling feature resilience testing results with results from the explainability tools facilitates a better understanding of the model's decisions. For example, char_freq_ $ with a high SHAP importance score may also have a high $\Delta F1$ when removed, indicating that the feature is incidentally correlated with other signals. On the other hand, a feature that is ranked high in the importance metric but has a low $\Delta F1$ suggests that the feature's impact is overestimated, meaning that the other features in the dataset that do not perform within the zeroed-out perimeter are sufficient.

This method, in particular, helps mitigate the challenge posed by adversarial spam and helps develop more robust classifiers by understanding how each model reacts to missing key attributes. This type of evaluation helps complement the explanatory and calibration approaches discussed in the other sections, aiming toward a single point of reference in tackling spam detection approaches that are efficient yet robust to drastic changes in spam techniques.

3.10 Implementation Details

Even though the selection of libraries or computational platforms is unlikely to affect the classification results, it will likely influence productivity for development, testing, and deployment. This subsection describes the policy principles underlying the methodology,

which aim to guarantee that every step, including data preprocessing and evaluation, is auditable and reproducible.

The primary language for coding in this work is Python because of the numerous libraries available for machine learning and data science. Two libraries form the backbone of the methodology have been utilized:

- Sci-kit-learn: A go-to library for classical ML models (e.g., Random Forest, Gradient Boosting, classical MLP) and utilities like feature scaling, train-test splitting, cross-validation, and grid/random hyperparameter search.
- TensorFlow/Keras: Used for building and training the implemented deep learning model. Keras allows effortless specification of a feed-forward layer with dropout and batch normalization. TensorFlow provides GPU support and sophisticated optimization routines as the backend.

The additional libraries NumPy and pandas take care of numerical calculations and data operations. Visualization activities involving calibration curves and correlation heat maps are done using Matplotlib and Seaborn. For model explanations and robustness checks, SHAP dissects model outputs to quantify the contributions of different features, and custom scripts take care of feature ablation by zeroing out or removing features to assess performance drop.

3.11 Hardware and Computational Environment

Although the Spambase set is mid-sized (~4601 emails and 57 numeric features), GPU can help speed up the training of Keras-based models, especially during hyperparameter searches or calibration phases. For most experiments, training on a single GPU is sufficient

to cut down the epoch times and allow more cycles of design choices (sometimes referred to as layer sizing or dropout sizing) to be made more frequently.

Tree-based models such as Random Forest and Gradient Boosting are usually run on CPU because they are much more efficient in taking advantage of multi-core parallelism than GPU processing. This environment combines the capabilities of both CPU and GPU, which helps with efficient training and tuning of each model.

Reproducibility remains a key concern for effective machine learning research, particularly with something as sensitive as spam detection, where changes, even slight, in random seeds or data splits can lead to entirely different outcomes. To alleviate the issue, the methodology implements Fixed Random Seeds. TensorFlow/Keras and sci-kit-learn have means to set random seeds for operations like shuffling, weight initializing, divide and conquer, and many other processes. Setting the same seed will ensure consistent results across different runs.

The other consideration is the Automated Scripts, as the execution of data cleaning, script model training, and calibration, along with resilience tests, is carried out via Python scripts. By distinctively defining each step, it becomes easy to copy or alter an experiment and its steps without unintentionally changing how it is executed. Also, using the Stratified K-Fold Cross Validation during the cross-validation process, as every fold is created through a specific random seed, ensures that the splitting is the same for different runs.

This is important to note due to the moderate class imbalance (around 60% ham and 40% spam). Even separately coded calibration and resilience tests are possible; having them all in one pipeline enhances consistency. After a model is trained and validated

through cross-validation, the pipeline executes temperature scaling for the deep learning model. It computes the calibration curves to check if there is any improvement. Then, the model feature resilience is evaluated by turning off certain features selectively (top-ranked versus randomly selected) and checking the model performance. This method ensures that tuning the model sensitivity to some features does not mask sensitivity to other features.

Each stage is logged, including features such as the original F1 score, the calibrated probabilities, and the performance drop with the feature being removed, which aids in understanding interactions among these procedures further.

Finally, the following flowchart illustrates the spam detection methodology utilized in this study. In the next chapters, further details concerning spam detection, the analysis results, and the outputs of the models used will be explained sequentially.

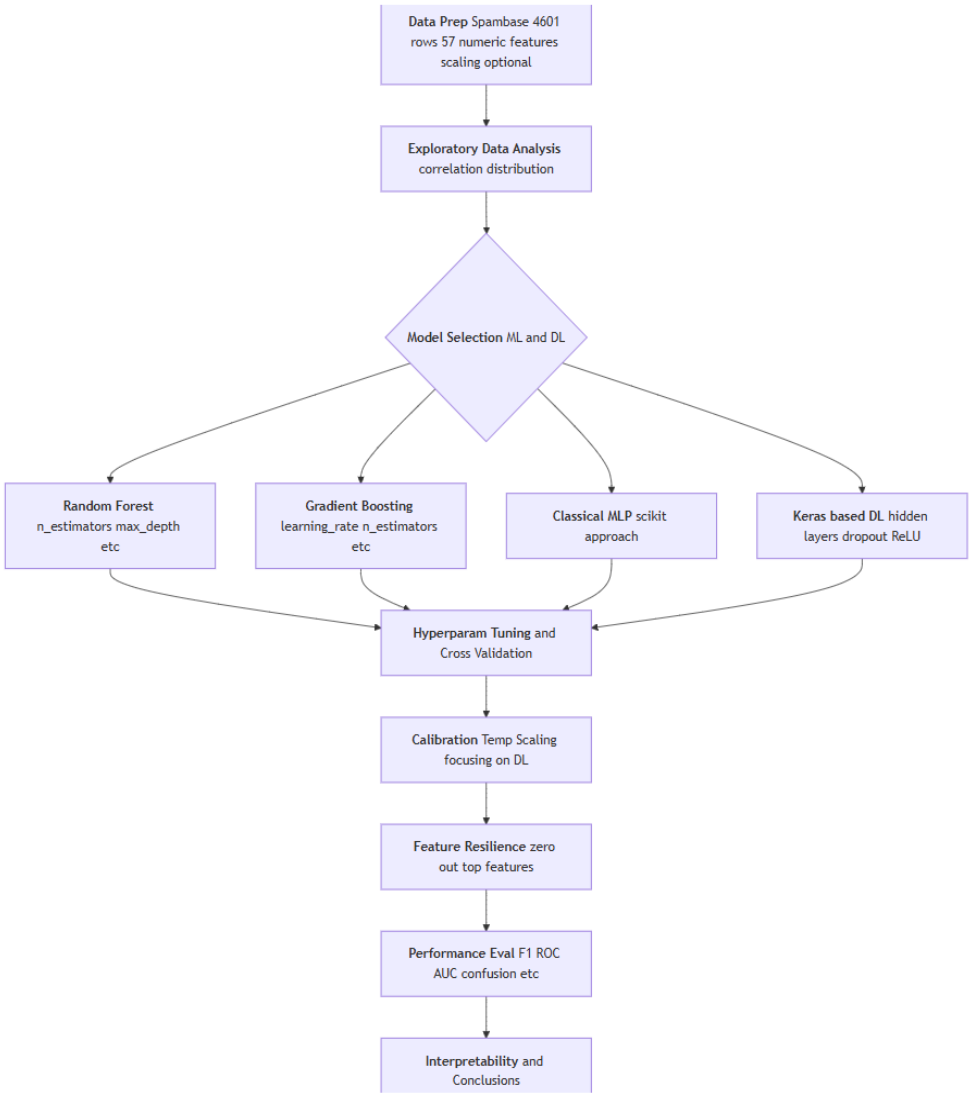


Figure 7: Overall Methodology Pipeline

Chapter 4: Results

4.1 Introduction to the Results Chapter

This chapter describes clearly how the empirical results from the spam detection process described in Chapter 3 are completed. At a high level, each selected model's goal shall be met: Random Forest, Gradient Boosting, a classical MLP, and a Keras-based deep learning model. The models shall be evaluated on the Spambase dataset on various parameters including, but not limited to, classification accuracy, F1-score, and ROC-AUC.

They will also include more elaborate evaluation aspects like probability calibration, feature engineering elegance, and explainability through SHAP analysis. Spam detection in real-life contexts involves more than accurate detection. This chapter deals with how well the model's calibration and predicted probability of spam detection. Also explained is how the model's response to feature removal (resilience) and which features-numeric, words, or character frequencies have the most substantial direct effect on the classification (interpretability). With this structure, it is possible to determine what model performs best according to the metrics, but it also enables an understanding of how robust and transparent the results were when used for live email filtering.

4.2 Initial Exploratory Findings

Before training the model, including the hyperparameter tuning process, a robust exploratory data analysis (EDA) was conducted to understand the relationships between the numeric features of the Spambase dataset and the spam label. This study built upon the correlation checking done in Chapter 3 by incorporating more detailed descriptive statistics

and visual tools, emphasizing the classification decision outlier or skewed distribution impacts. Ultimately, the findings aimed to verify whether certain numeric signals, such as the frequencies of words, punctuation, and capitalization, did, in fact, constitute strong, verifiable indicators for spam detection, as was suggested in prior chapters.

One of the principal tasks of this study was to analyze specific data features that may be important for guiding future modeling work. In Figure 8 (Correlation Heatmap for Top 15 Features), a selection of the most correlated attributes and their interactions with the spam label are represented. These features consist of various lexical features such as the frequency of certain words like ‘remove,' ‘000’, and ‘business,’ character features such as ‘!’ and ‘$,’ and capital letter features like ‘total run the length of capital letters’ and ‘longest run of capital letters’.

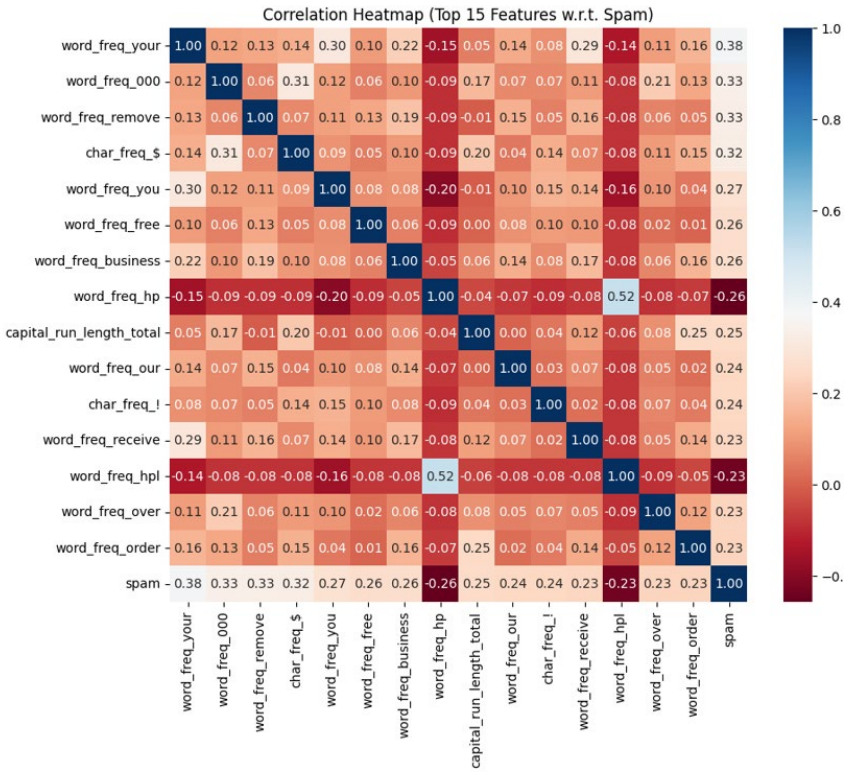


Figure 8: Correlation Heatmap for Top 15 Features

The scale goes from the deepest red, signifying strong positive correlations, to the deepest blue, indicating negative correlations and medium colors represent moderate correlations.

For detailed definitions of all study variables, please refer to the data dictionary provided in Appendix Section A.1

The heatmap indicates that specific lexical indicators, such as word_freq_your, word_freq_remove, and word_freq_000, exhibit a moderate to strong positive correlation with the variable spam. This further supports the assumption that spam emails usually track promotional or finance-related words. Character-based features, such as char_freq_! and char_freq_$, also show a positive correlation, suggesting exclamation marks and dollar signs in suspicious messages. Of note, there are positive correlations with some metrics of capitalization, such as capital_run_length_total, which indicates the number of uppercase letters in text blocks. This suggests that blocks of upper-case letters and components of unpleasant content may dominate trolls' texts. All of these observations, taken together, suggest confirming the numeric encoding of textual signals and indicating which features will be most important and valuable in later classification and interpretability analyses.

The EDA analysis included whether any pair of features attained very high correlation values, such as above 0.90, because that would mean one could be eliminated without losing significance, thus improving ease of interpretation. In this dataset, such pairs were not found, indicating that while useful redundancy exists, like some lexical features having a moderate correlation with punctuation, it does not seem that these features are strong enough to endanger the stability of the modeling pipeline. Instead, these partial

correlations point out that some spammers employ multiple tactics. These insights particularly shaped the reasoning for the next methodological steps.

First, since no single characteristic completely rules the spam–ham domain, the study kept all 57 numeric attributes to avoid feature covering's preemptive strike. Second, recognizing that specific algorithms' (such as the feedforward neural networks) standard scaling would tend to use very high values in char_freq_! or capital_run_length_total overshadowing other less strong signals raised the possibility of applying them to some of the skewed distributions. Third, the partial overlaps among top features led to the integration of multiple cues, which resulted in the classification. For instance, without capital_run_length_total (a high-correlation feature), the model could be disabled.

However, because other signals like char_freq_! or word_freq_remove can compensate, the model is not bound to be disabled. The extended EDA confirmed that the numeric representation of Spambase does indeed encapsulate various spam-related textual and stylistic features. Features related to promotional language, punctuation intensity, and use of uppercase letters are essential in distinguishing spam from non-spam messages in the dataset. These conclusions were directly used to inform the next stages of model choice, model parameterization, and model robustness verification, which constructs a rationale to treat the signals given numerically by the dataset as the main predicted features instead of mere bystander variables.

The mean values of various SHAP feature scores are depicted in the bar graph below. These scores are identically averaged and given in descending order, illustrating which features explain the variance in the model's spam likelihood the most.

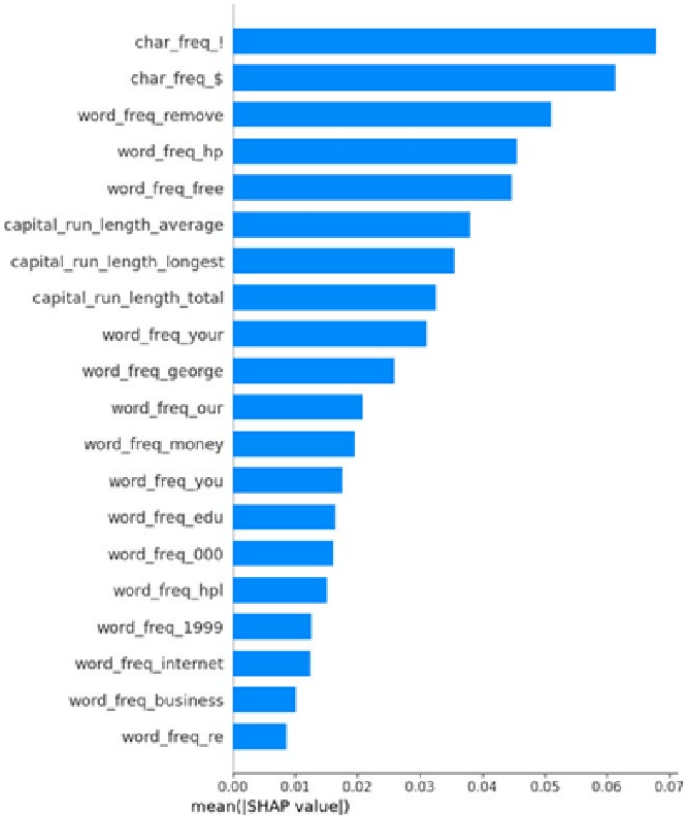


Figure 9: SHAP Bar Plot for Feature Importance

This tells us that every feature, in this case, punctuation or word usage, will have a SHAP score or influence depending on how much that specific feature was utilized across all emails in the test set. The highest mean absolute SHAP value at the top of the list is char_freq_!, which emphasizes the model's reliance on exclamation marks as a strong spam identifier. Spammers often use several exclamation marks to get attention.

The second feature, char_freq_$, also relates to dollar signs or references to money, typical of spam messages that seek to advertise some financial boon or ask for money. Just below, word_freq_remove shows how Terms, like remove or unsubscribe, are also found to considerably increase the chances of spam, given the large proportion of unsolicited messages analyzed.

The presence of word_freq_hp, word_freq_free, and multiple “capital_run_length” reflect how spam filters combine the lexical and the stylistic. The repeated use of the term “free” or blocks of upper-case letters of the text are frequently used for advertising or fraudulent purposes, thus causing the model to assign more outstanding value to those features for spam classification. Others, like word_freq_george or word_freq_our, as having more specific low frequencies may appear strange, but they stem from identified patterns during training, possibly because spammers had specific names or phrases that were so frequently used that the model associated higher levels of spam with them.

Moving down the list features such as word_freq_money, word_freq_your, word_freq_edu, and word_freq_000 have moderate mean SHAP values. They are less important than exclamation marks and dollar signs but continue to be influential. Even lower-ranked features, such as word_freq_hpl, word_freq_1999, and word_freq_meeting, have even less impact by themselves, but when combined with other signals, can push borderline messages to be classified as spam or ham. Collectively, these features illustrate that the model does not rely on a single numeric dimension. It combines punctuation, word frequency, and capitalization to form firm spam likelihood estimates. Practically, this SHAP bar plot is another aid for administrators and researchers, allowing them to pinpoint which numeric attributes are most important for routine filtering.

Knowing that exclamation marks, dollar signs, and certain lexical items are responsible for classifying spam allows them to track changing spam strategies (for instance, shifts in behaviors like reducing exclamation marks or changing the word “free” to its synonyms) and make predictions for filter calibration. Also, this supports the feature resilience results: removing one of the top features, in this case, char_freq_! It will likely

cause a modest dip in performance, but the model's feature dependency structure ensures that spammers cannot easily bypass the system by changing one cue.

4.3 Baseline Performance of Each Model

This part analyzes the results of each classifier—Random Forest, Neural network Gradient Boosting, classical MLP, and Keras Deep Learning model—with respect to the Spambase dataset without any prior re-calibrations or feature engineering. The results are from the held-out test set after stratified k-fold cross-validation and tuning all the hyperparameters for each model. The accuracy, F1 score, and ROC AUC are the primary metrics achieved, along with confusion matrices illustrating each model's true/false positive and negative outcomes.

4.3.1 Models' Classification Report

Model	Label	Precision	Recall	F1-Score	Support
RandomForest	0	0.9424	0.9677	0.9549	558
	1	0.9483	0.9091	0.9283	363
	accuracy			**0.9446**	921
	macro avg	0.9453	0.9384	0.9416	921
	weighted avg	0.9447	0.9446	0.9444	921
GradientBoosting	0	0.9404	0.9606	0.9504	558
	1	0.9373	0.9063	0.9216	363
	accuracy			**0.9392**	921
	macro avg	0.9388	0.9335	0.936	921
	weighted avg	0.9392	0.9392	0.939	921
MLP	0	0.9397	0.9498	0.9447	558
	1	0.9216	0.9063	0.9139	363
	accuracy			**0.9327**	921
	macro avg	0.9306	0.9281	0.9293	921
	weighted avg	0.9326	0.9327	0.9327	921
DeepLearning	0	0.9445	0.9462	0.9454	558
	1	0.9171	0.9146	0.9159	363
	accuracy			**0.9338**	921
	macro avg	0.9308	0.9304	0.9306	921
	weighted avg	0.9337	0.9338	0.9338	921

Figure 10 : Models' Classification Report

4.3.2 Analysis of Random Forest

These results demonstrate the operation of a spam filter that captures unwanted emails with few false positives, which is a desired goal. Looking at Class 0 (Ham), the precision value of 0.94 indicates that 94.24% of all messages identified as ham are ham, and the recall value of ~0.97 indicates that the model can identify 96.77% of the ham messages. Together, they produce an F1 score of 0.95, indicating that the amount of spam the model blocks is very low. Class 1, or the spam category, produces precision scores of ~0.95, which indicates that when the random forest model predicts a spam message, it is 94.83% of the time.

The model also has a recall of ~0.91, which means that it captures approximately 90.91% of the spam messages in the dataset. With the spam messages not perfecting coverage, the F1 score of ~0.93 indicates that the model makes a modest, rough estimate of 9% of spam messages getting through.

In aggregate terms, an accuracy of 0.94 proves that the model correctly classifies 94.46% of the entire test set. The acro avg (F1) of 0.94 shows that the algorithm treats spam and non-spam equally and can also use them all reasonably well. The weighted average (F1) of 0.94 spans proves that the ham tends to the more significant data proportion so that when the model is set with the weight of the overwhelming class, the spam filters still heavily dominate them. Random forest offers maximum accuracy by not misclassifying captured ham that would annoy users unnecessarily while capturing most spam messages, which definitely increases the chances of granting minimally excessive data damage.

Moderators could implement boundary changes to improve spam capture to above 91%. However, that would increase the chances of indiscriminate targeting and lower precision. In conclusion, these data suggest that the Random Forest model has the best ability to detect spam numerically compared to other methods studied in this paper.

4.3.3 Examination of Gradient Boosting

The metrics achieved through Gradient Boosting are equally impressive, if not slightly lower than those achieved through Random Forests. For Class 0 (Ham), the reported precision is 0.94, meaning that out of all the emails marked as ham, 94.04% are accurate, and the recall of 0.96 shows that it can successfully identify 96.06% of all legitimate messages. This yields an F1-score of 0.95, which indicates that the model is highly proficient at reducing false positives to ham. In Class 1 (Spam), the precision stands at ~0.94, which means that out of all flagged spam, 93.73% is spam, and the recall of 0.90 indicates that it captures approximately 90.63% of all spam messages. The obtained F1-score of 0.92 indicates robust coverage from Gradient Boosting for spam.

However, there is a concession to the ~0.93 achieved by Random Forests for spam detection. With an accuracy of ~0.94, it can still be contended that 93.92% of all emails are classified correctly, which lags Random Forest by almost half a percentage.

The Macro average (F1) of 0.93 demonstrates that class balance is achieved from both sides. In contrast, the Weighted average (F1) of ~0.94 demonstrates that taking into account greater support for the classification of ham, the model can still be classified firmly, demonstrating the overall classification ability of the model. In practicality, the slight drop

of Gradient Boosting compared to the random forest is not likely to have an effect on real-world deployment, as these differences equal a few messages in filtering on a large scale.

Some system administrators prefer the boosting approach because it is iterative and adjustable by the learning rate and number of boosting rounds, allowing it to better account for changing spam patterns over time.

4.3.4 Analysis of MLP (Classical Neural Network)

For the MLP classifier, the last report indicates that Class 0 (Ham) has achieved a precision of ~0.94, recall of ~0.95, and F1 score of 0.94 while having a support of 558.

This means that for all emails labeled as ham, approximately 93.97% are legitimate, and this model captures approximately 94.98% of all ham messages. The resulting F1 score of 0.94 reveals how well the model captures precision and recall for the majority class, which is quite positive. On the other hand, looking at Class 1 (Spam), the MLP obtains a precision value of 0.92 and a recall of ~0.91, leading to an F1 score of 0.91 at 363 supports.

Although the model lets about 9 to 10% of spam messages through, it captures many spam messages well. The model's overall accuracy sits at 0.93, the macro average F1 score is about ~0.93, and the weighted average F1 score is around 0.93 as well. These results show that the MLP does well in numeric-based classification, but as is known, it does worse than the ensemble methods of higher order.

The MLP’s architecture appeals to Administrators because it is much simpler, mainly if threshold tuning or additional calibration steps for borderline messages are

applied. The ham recall being close to 0.95 also ensures that while there are almost no disruptions to legitimate email, the traffic is positively identified.

4.3.5 Analysis of the Deep Learning (Keras-Based Model)

In the Keras-based Deep Learning classifier, Class 0 (Ham) has a precision rate of 0.94, a recall rate of ~0.95, and an F1-score of 0.94. These scores correspond to the same support of 558 ham messages in the test set. This means the model is correct approximately 94.45% of the time when predicting an email as spam and can correctly recognize about 94.62% of all identified messages. With Class 1 (Spam), the network reports a precision of ~0.92 and recall of 0.91, with an F1-score of ~0.92 and support for 363 spam emails. It is certainly not as high as some ensemble approaches, but it is still quite powerful, enabling the capturing of almost all spam. Accurately, this translates to 0.93, meaning that 93.38% of all emails (ham and spam) are classified correctly. The macro average F1 is roughly 0.93, while the weighted average F1 is around 0.93 demonstrating the model's performance with the class distribution of the dataset. These measures suggest the deep architecture outperforms the MLP, implying it is superior in capturing intricate numeric interactions of word frequencies, usage of punctuation, and capitalization metrics.

Nonetheless, the methodology still lags behind ensemble approaches regarding accuracy and spam recall, suggesting that more tuning or domain-specific embeddings may be required to close the performance gap. Administrators who prefer the flexibility offered by neural networks may be particularly attracted to this solution, especially when used in conjunction with sophisticated adjustments such as temperature scaling, which reconciles predicted probabilities with actual spam likelihood.

4.4 Consolidated Comparison and Observations

Analyzing the classification reports in conjunction with the metrics of accuracy, macro average F1, and weighted average F1 reveals clear patterns that apply to all four Random Forest, Gradient Boosting, MLP, and Deep Learning classifiers.

Figure 11 : Comparison of Model metrics

In summary, Random Forest comes out on top, Gradient Boosting comes next, and the two neural models are tightly clustered just below the ensembles. All four methods achieve a remarkable balance between ham and spam, so they rarely block genuine mail while correctly flagging most spam. From a practical perspective, the minor differentiations among them might compel some decision-makers to focus on computational cost, preferred level of explanation, and convenience of hyperparameter optimization. No matter the decision made, the general metrics support that utilizing various modeling approaches, the

numeric form of textual attributes, word occurrences, punctuation usage, and capitalization metrics results in spam filtering effectiveness across the board.

4.5 Analysis of The Confusion Matrix for All Four Models

The four confusion matrices illustrated below help analyze how each classifier—Random Forest, Gradient Boosting, MLP, and Deep Learning—allocates its accurate and inaccurate predictions among the ham (class 0) and spam (class 1) classes.

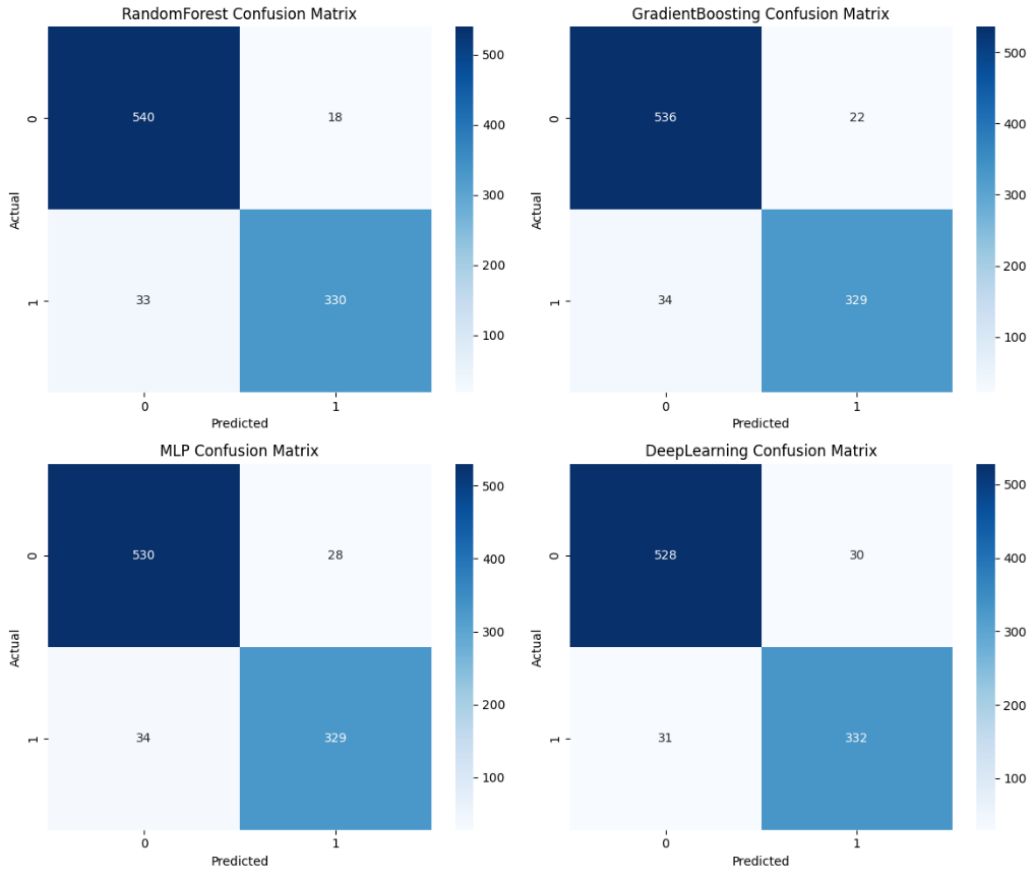


Figure 12: Confusion matrices of the four models

In all matrices, the true negatives (correctly classified ham) and false positives (ham that was misclassified as spam) are shown in the top-left cell versus the top-right cell, respectively. While all models demonstrate excellent accuracy in numerical spam detection, each matrix highlights how the performers balance the tradeoff between ham and spam regarding the capture, reveal, and misclassification.

Random Forest is exceptional in having the least total misclassifications, achieving 540 correct identifications of ham with only 18 false positives and correctly identifying 330 spam and missing 33 spam. This elegance indicates that the model can save legitimate emails from quarantine while capturing the most unwanted emails.

Gradient Boosting trails closely behind, accurately identifying 536 ham and 329 spam and having 22 false positives and 31 false negatives. Although it concedes a handful more misclassifications than Random Forest, it is still very trustworthy since it mostly captures only a tiny portion of the spam it misses.

The MLP shows 530 true negatives and 28 false positives for ham, which is accompanied by 329 correctly identified spam and 34 missed spam emails. These numbers suggest that the MLP underclocks or overestimates the number of spam messages by a smaller margin than the ensemble methods. Likewise, the Deep Learning model records 528 true negatives, 30 false positives, 31 false negatives, and 332 true positives. Even though it labeled a little more ham as spam relative to the ensembles, it is capturing more spam emails than the MLP, demonstrating the stable differences in how the model's neural layers parse punctuation, word frequencies, and capitalization cues.

This section focuses on the matrices where the combined confusion matrices show that all four models maintain a strong balance between ham and spam classification. Still, the ensemble approaches almost always reduce errors more efficiently. Neural approaches, especially deep networks, tend to overly limit the number of ham emails marked or let through a small fraction of spam. However, the large true-positive values make them suitable for filtering tasks in practice. For example, Random Forest or Gradient Boosting allows administrators who want to minimize damage to ham to pursue those solutions, while for more risk acceptance scenarios, neural methods can get decent coverage with well-set thresholds or calibration.

4.6 ROC Curves Comparison for All Models

The curves of the ROC shown in this figure below represent the FPR or False Positive Rate, which is placed along the X-axis, while the TPR or True Positive Rate is put along the Y-axis for all classifiers (Random Forest, Gradient Boosting, MLP, Deep Leaning classifier) with every change of set decision mark.

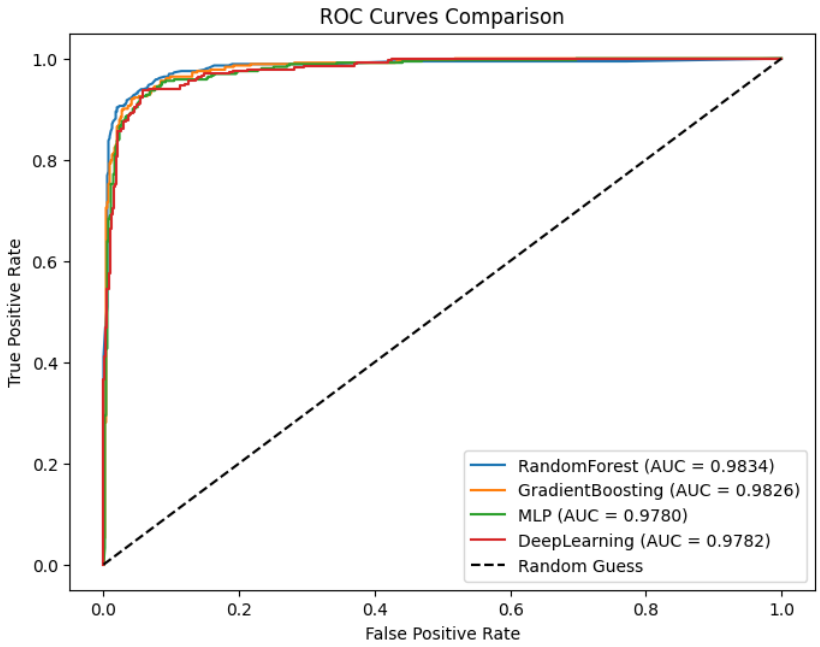


Figure 13: All Models' Curves Comparison

The four models in the numeric-based spam detection study show ROC curves resembling the graph's upper-left corner; the Random Guess baseline, which classifies emails to spam or ham randomly, is represented by a dotted diagonal line. All models in the study demonstrate strong discrimination abilities. While this strong discrimination ability shows, there are still some differences in AUC value, which suggests that there could be small yet noticeable performance differences. Random Forest has the most substantial AUC value of ~0.98 among all four. In the top left, there was a clear positive correlation, which means the model is good at capturing most spam messages without much legitimate mail being misclassified, making the curve so high in that area. Gradient Boosting is the other model with slightly less AUC at ~0.98, and the line still follows the step shape of Random Forest but diverges a bit in the mid-FPR area. The boosting's error correction in the boosting strategy of mid-levels of strong spam-ham signals heuristically

produces an almost average result in final performance measures with Random Forest, resulting in a numeric signal identification gauge lower than the forest. MLP and Deep Learning log AUC scores of approximately 0.98. Even though their curves are close to the top-left quadrant, they are slightly lower than the ensemble methods.

This difference usually manifests around the mid to high TPR areas where the neural approaches are likely to give up a bit more TPR at a given FPR or, on the contrary, have to accept a somewhat lower TPR at a given FPR. Regarding actual spam filtering, these differences translate to a few more false positives or "missed" spam filters at certain threshold levels. The two neural networks are still well above the random estimate value, proving their ability to predict is excellent.

In AUC terms, the difference between the best model (Random Forest at ~0.98) and the neural models is not large for practical use, but it does form choices that are made based on thresholds. In an organization whose email guard solution aims to capture almost all spam while minimizing false positives, ensemble methods, particularly Random Forest, will likely offer a higher TPR for the same FPR. On the other hand, MLP or Deep Learning models may only need slight modifications of the thresholds to achieve that level of performance, particularly in the vicinity of borderline values. Still, all four models perform exceptionally well in spam detection based on numbers, as indicated by the ROC curves.

Each curve goes well beyond the diagonal reference line. Random Forest and Gradient Boosting still have an edge, albeit small, over the rest, but the neural models are not far behind as they can frequently be adjusted or adapted to bridge that gap.

4.7 Precision-Recall Curves Comparison for All Models

The variability in the decision threshold for each classifier – Deep Learning, Random Forest, Gradient Boosting, and MLP – is demonstrated by a variation in the precision-recall relationship shown in these four plots.

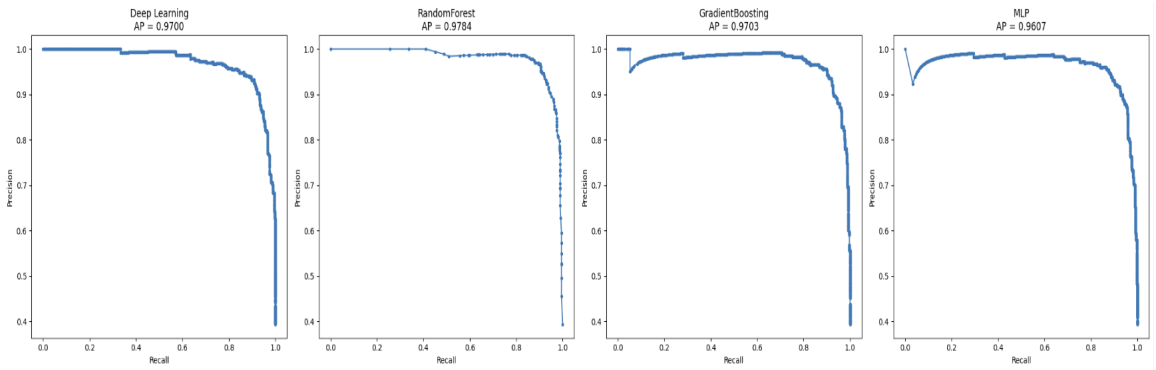


Figure 14: Comparison of All Models' Precision-Recall Curves

.

Unlike ROC curves, which evaluate the proportion of true positive and true negative instances relative to all positive and negative cases, precision and recall curves are particularly useful in cases where the positive class (spam) occurs infrequently yet is exceedingly important. The AP (Average Precision) values illustrated above each curve provide an overarching figure describing how well each model sustains elevated precision for high recall values.

The Deep Learning model (the leftmost plot) rakes in an AP of approximately 0.97, meaning it can maintain reasonably high precision, frequently over 90%, until recall begins to approach 0.90 or more. This suggests the network can identify most spam with a reasonable amount of false positives, although the curve may drop steeply after high recall

levels. Random Forest (second from the left), with an AP around 0.98 or higher, is the best performer. Its curve stays closer to the top right corner for a longer section of the x-axis, meaning that for a given recall, it has higher precision than the other models during that portion. This benefit further supports the capability of Random Forest to mark spam emails while misclassifying legitimate emails efficiently.

The approximate value of 0.97 for AP places Gradient Boosting (third from the left) running nearly close to the Deep Learning model. Its curve follows the Random Forest's shape early in the recall range. On the other hand, it begins to decline towards the end of the dip in recall above 0.90. Still, the iterative boosting process that sharpens clearly defined cases for Gradient Boosting to work on allows the model to sustain strong precision until it peeks into the upper recall area. MLP (rightmost plot), an AP of almost 0.96, also does not perform quite well on the high precision sustainment at high recall compared to the others. Although strong overall and clearly above any random guess baseline, the model is likelier to exhibit a sharp drop in precision at about 0.90 or above recall. This behavior indicates a simpler feedforward architecture lacking detailed feature numeric weighting present in the ensemble or deep neural networks, leading to mislabeling some more legitimate emails as spam at higher recall rates.

These precision-recall curves assist users in picking out thresholds that correspond with organizational goals, like reducing false positives or maximizing the capture of spam. The Random Forests model has the highest spam precision and hugest boost in precision, which means it has the greatest increase in spam coverage and precision.

Gradient Boosting follows in close order, while Deep Learning and MLP remain reasonable alternatives but suffer sharper precision drop-offs at high recall. Overall, the

four models outperform a set of naive baselines and perform well on discrimination tests. The thin differences in their precision-recall curves help make threshold and calibration decisions in spam filtering solutions.

4.8 Deep Learning Model - Loss and Accuracy Learning Curves

The two plots below show the progression of training and validation metrics over Keras-based Deep Learning model epochs for a feedforward network trained on spam numerical features and offer a glance at how a spam-blocking model learns over time.

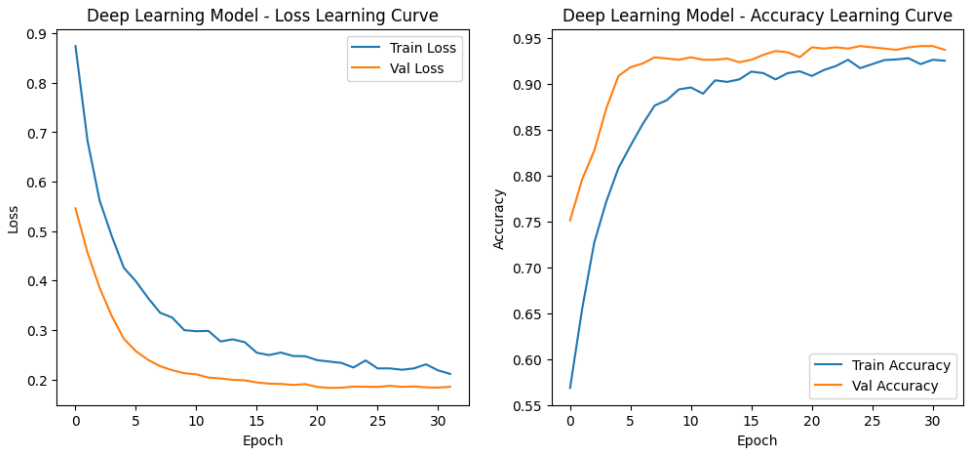


Figure 15: Deep Learning Model - Loss and Accuracy Learning Curves

The left shows loss (typically some form of binary cross-entropy), and the right shows accuracy (how many out of all the emails yelled as spam were spam). Together, they indicate whether the model overfits the training data.

In the left figure's loss curve, the blue line indicates that the training set begins at a relatively high loss value when the network's weights are assigned randomly or partially optimized. During the first few epochs, the training loss lowers steeply, implying that the

model is learning from word frequencies, punctuation, and capitalization that differentiates spam from ham. In the same period, the orange line representing validation loss is approximately constant and consistently lower than the training loss curve. This steady decrease is a clear sign that the network is memorizing training examples and learning patterns that can be applied to new data. If the validation loss stagnated quickly or diverged greatly from the training loss curve, it would be a sign of overfitting. It is necessary to apply stronger regularization or reduce the number of epochs.

The right graph shows the accuracy curve, which reflects these patterns in performance. The training accuracy blue line shot up steeply from an initial low value, surpassing 0.90 after several epochs when the network adjusted its internal weights to identify spam signals. The orange line for validation accuracy stays slightly below and, at other times, slightly above the training curve, eventually settling in the mid-0.90 range (typically 0.93 to 0.95) in the final epochs. The validation accuracy converging closely to the training accuracy indicates the model's reasonable skill in dealing with the numeric features, including char_freq_! word_freq_free, or capital_run_length_total, across the entire data distribution instead of solely relying on the training set's peculiarities.

These curves determine the required number of training epochs, the timing of early stopping, and the possibility of achieving better results with new layers or higher dropout rates. For example, if the training accuracy kept increasing but the validation accuracy started to level off, this could indicate overfitting that would need an earlier stop or change in hyperparameter strategy. However, the smooth convergence depicted in these curves suggests that the architecture and training schedule are well-suited for the network's ability to encompass the multi-dimensional numeric indicators of spam detection without

overfitting. In summary, these learning curves provide further support and confidence in the relevance of a moderately deep feed-forward model for spam filtering of numeric type located between simple MLPs and powerful ensemble systems.

4.9 Outcomes for Calibration of Probabilities

In practice, spam detection does not rely solely on the classification performance of a model but also on how well it estimates the odds of an email being spam. Cybersecurity administrators usually estimate these probabilities to decide whether messages should be blocked, quarantined, or delivered, especially for borderline cases where numeric cues can be vague. A highly accurate classifier can, under poor calibration of predicted probabilities to actual spam frequencies, block legitimate messages or allow spam through. This section looks at the ensemble and MLP calibration curves and the calibration features of the Deep Learning model's temperature scaling augmentation with bin-based tables showing the probability estimates of spam occurrences and the actual spam rate.

4.9.1 Calibration Observations for Ensemble Methods and MLP

The three figures below tell how the mean predicted probability of each model differs from the fraction of positive cases (actual spam rate) for the selected probability bins.

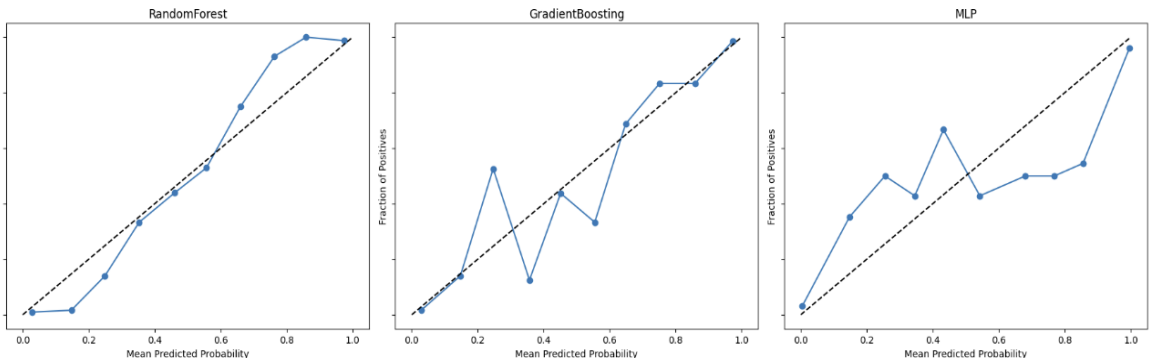


Figure 16: Calibration Curves for Random Forest, Gradient Boosting, and MLP

The curve displays the predicted probabilities of email bins with the average predicted probability on the x-axis and the true fraction of spam on the y-axis. The model's curve is, therefore, an illustration of email bins with predicted probabilities that fall within a specific range, for example, (0.1-0.2), (0.2-0.3), etc.

The random forest model (on the left side) demonstrates loose calibration and moderate overestimation. Mid-level probabilities of 40 to 80 percent particularly illustrate these tendencies. Random Forest's output probabilities indicate low trust, but security administrators can place spam threshold proportions, anticipating that about 70% of emails at or above the predicted probability are spam. As shown in the center panel, Gradient Boosting has a reasonable fit along the diagonal but tends to display slight overconfidence or under confidence in some of the bins.

For example, the curve appears higher than the diagonal for mid to high-predicted probabilities. This means that the model will likely overestimate the likelihood of spam in these ranges. On the other hand, the model underestimates spam rates when the curve is lower than the diagonal in the lower bins. In any case, the overall alignment is still

reasonably close because of the moderate calibration probabilities achieved from the iterative refinement of numeric cues (for example, punctuation usage and word frequencies) with Gradient Boosting. These points could be made closer to the diagonal with minor re-calibration or threshold tuning. However, even so, the default output is acceptable for many real-world filtering applications.

The MLP (right panel) seems more unpredictable, with some bins coming close to the diagonal and some deviating by several percentage points. This flexibility might capture the simpler feedforward topology's pathological under- or overconfidence in certain numeric intervals (e.g., borderline exclamatory sentence usage or runs of uppercase letters). While the overall shape still suggests a reasonably well-calibrated model, particularly in higher probability bins where spam is more abundant, it might be the case that administrators need to apply some form of post-processing, such as temperature scaling, if they are too reliant on accurate probability values. Regardless, the near diagonal trend of the MLP indicates that a baseline level of probability output, though far from ideal, is usable for making reasonable decision thresholds.

These self-explanatory calibration curves show that Random Forest and Gradient Boosting attempts at estimating probabilities are relatively accurate, as they almost perfectly follow the diagonal. The MLP has similar estimates but is liable to exhibit more drastic changes from bin to bin, possibly needing less smoothing if the organization calls for higher precision. These deviations aside, the calibration curves above a random baseline confirm the smoothed models outperform estimation spam detection systems, where system administrators are expected to depend on the predicted probability of spam to reduce the number of false positives and false negatives.

4.9.2 Deep Learning Model Temperature Scaling

Neural networks, especially deeper ones, do not self-calibrate well to produce probability outputs. The Keras-based Deep Learning model had a glaring problem with the predicted spam probabilities relative to the spam frequencies. To overcome this problem, the model employed calibration in the form of temperature scaling, a process where a scalar T is applied to either lower or raise the model's logits output prior to the final sigmoid execution.

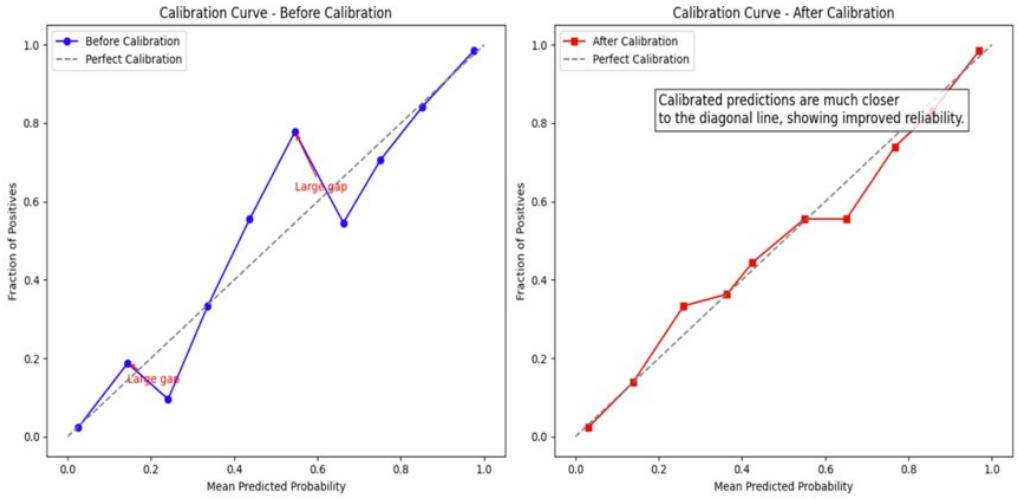


Figure 17: Calibration Curves for Deep Learning Model

These two plots illustrate how post-processing calibration, specifically the application of temperature scaling, improves the Deep Learning model's probability estimates for spam detection. The left side curve shows the model's calibration prior to applying temperature scaling. The right side of the same curve shows the post-calibration results. Each point on these curves is representative of a bin of emails corresponding to a specific predicted probability range (e.g., 0.0 to 0.1, or 0.1 to 0.2, etc.). The x values

represent the mean predicted probability of positive responses to those emails, while the y values represent the actual rate of positive responses (spam) observed in each bin. The dotted diagonal indicates

In the pre-calibration graph on the left, the curve is clearly shifted away from the line, especially for middle probabilities (0.4-0.8). Many bins are relatively high or low to the line, meaning the model's confidence in borderline emails does not always match the actual spam probability. For example, a bin with a predicted probability of 0.6 may actually have a spam fraction of 0.4 (underestimation) or 0.7 (overestimation). Such errors can produce inadequate thresholding in decision-making because users may naively believe the predicted probability corresponds to the actual spam probability.

On the other hand, the right-side plot demonstrates a curve closer to the diagonal, suggesting that the temperature scaling step calibrated the model probabilities appropriately. Previously, the miscalibrated bins now have higher proportions of spam than predicted and lower proportions than observed, moving towards the line. Thus, a bin with a mean predicted probability of 0.7 now corresponds to 70% spam, which is extremely helpful in thresholding scenarios that need to minimize the cost of false attacks (legitimate mail gets blocked) and miss (spam gets overlooked). This enhanced alignment is most dramatic in the 0.5-0.8 range, where the model had actually overestimated most of its errors before calibration.

From practice, cybersecurity administrators can rely on adjusted probabilities more confidently while setting cutoffs for these post-calibration curves. Suppose they set the automatic quarantine cutoff of 0.85 spam likelihood. In that case, they can trust that a little under 85 percent of the emails are spam rather than hope that the probability is not too

exaggerated or depressed. Therefore, calibration increases the valuable value of the model, although it does not alter the calibration's raw accuracy and F1 scores on the comments.

The model's output closely illustrates the actual spam distributions, adding an important level of robustness needed for threshold-based filtering. This is thanks to the reduced borderline misclassifications that there is calibration.

Below, the table illustrates how post-processing calibration—in this case, temperature scaling—alters the likely spam rates forwarded to the Deep Learning model to better reflect the actual spam rates.

Bin	Mean Predicted Prob (Before)	Fraction Positives (Before)	Mean Predicted Prob (After)	Fraction Positives (After)	Gap Before	Gap After
1	0.0254414	0.0234542	0.0314813	0.0238807	0.0019872	0.0076006
2	0.144162	0.1875	0.139166	0.139535	0.043338	0.000369
3	0.241169	0.0952381	0.259735	0.333333	0.145931	0.073598
4	0.336279	0.333333	0.363936	0.363636	0.002946	0.0003
5	0.437122	0.555556	0.425311	0.444444	0.118434	0.019133
6	0.545685	0.777778	0.549978	0.555556	0.232093	0.005578
7	0.661974	0.545455	0.650727	0.555556	0.116519	0.095171
8	0.750852	0.705882	0.766761	0.73913	0.04497	0.027631
9	0.851273	0.84127	0.85705	0.830508	0.010003	0.026542
10	0.974837	0.984375	0.969433	0.984615	0.009538	0.015182

Figure 18: The combined calibration values

Each row relates to a bin of emails that are placed into cuts based on the predicted probability (e.g., 0.0–0.1, 0.1–0.2, etc.). The rows of real and assumed spam show the mean assumed probability and the predicted spam fraction (real spam) for this bin, both prior to and post-calibration, as well as the difference in value.

Before calibration, some bins show significant differences between the model's predicted average probabilities and the actual fraction of spam. For example, a bin may have a mean predicted probability of around 0.54 but have about 0.40 spam emails, meaning the model overpredicts the chances of spam at that level. On the other hand,

another bin may have a mean predicted probability of approximately 0.70 with an actual spam fraction of around 0.78 – illustrating the model's underprediction. Such gaps can make it challenging to make threshold-based decisions if the model users believe that the unprocessed estimates directly represent the actual spam proportions.

After temperature scaling, the predicted means probabilities associated with the positive outcomes bin are more likely to agree with the observed figures. The “Gap Before” and “Gap After” columns demonstrate each bin's differences before and after calibration.

Often, the large initial discrepant bins appear to undergo the most drastic changes, whilst those nearly aligned to the diagonal line (in typical calibration curves) exhibit little difference. The calibration improvement summary note (e.g., bins 2, 3, 5, 6, 7 show “improved calibration”) adjoined to the graphs highlights that midrange bins, where spam probability lies between 0.4-0.7, experience the most temperature scaling gain. In practice, borderline emails that result in these bins are most likely to be misclassified, so it is essential to ensure that predicted probabilities are congruent with tangible outcomes.

The predicted spam probabilities from the Deep Learning model become more reliable with this probability distribution achieved from temperature scaling. Email security administrators or solutions no longer need to work with skewed metrics as they can set a threshold (e.g., 0.70) with greater confidence that probability values at or above that figure truly possess a ~70% chance of being spam.

4.10 Feature Resilience Results

The strength of a model depends on how accurately it predicts spam and the trustworthiness of these estimates. However, robust performance is also defined by how

each model responds to removing or manipulating critical features. In argumentative contexts, spammers may avoid or manipulate the text signal that filters depend on the most.

In this case, feature resilience is explored by zeroing out or removing specified attributes and assessing the resulting classifier performance to determine dependency on the chosen numeric signals.

Feature	Model	Original F1	Modified F1	Delta F1
word_freq_make	Deep Learning	0.911323	0.914052	-0.00272851
word_freq_make	RandomForest	0.92827	0.925666	0.00260384
word_freq_make	GradientBoosting	0.921569	0.921569	0.0
word_freq_make	MLP	0.913889	0.91516	-0.00127106
word_freq_address	Deep Learning	0.911323	0.912568	-0.00124498
word_freq_address	RandomForest	0.92827	0.929577	-0.00130742
word_freq_address	GradientBoosting	0.921569	0.921569	0.0
word_freq_address	MLP	0.913889	0.910615	0.00327436
word_freq_all	Deep Learning	0.911323	0.911323	0.0
word_freq_all	RandomForest	0.92827	0.929775	-0.00150524
word_freq_all	GradientBoosting	0.921569	0.921569	0.0
word_freq_all	MLP	0.913889	0.921569	-0.00767974
word_freq_3d	Deep Learning	0.911323	0.912807	-0.00148321
word_freq_3d	RandomForest	0.92827	0.92827	0.0
word_freq_3d	GradientBoosting	0.921569	0.921569	0.0
word_freq_3d	MLP	0.913889	0.913889	0.0
word_freq_our	Deep Learning	0.911323	0.910082	0.00124158
word_freq_our	RandomForest	0.92827	0.922207	0.00606354
word_freq_our	GradientBoosting	0.921569	0.910638	0.0109303
word_freq_our	MLP	0.913889	0.906077	0.00781154
word_freq_over	Deep Learning	0.911323	0.911323	0.0
word_freq_over	RandomForest	0.92827	0.929577	-0.00130742
word_freq_over	GradientBoosting	0.921569	0.921569	0.0
word_freq_over	MLP	0.913889	0.915862	-0.00197318
word_freq_remove	Deep Learning	0.911323	0.90095	0.0103735
word_freq_remove	RandomForest	0.92827	0.918455	0.00981511
word_freq_remove	GradientBoosting	0.921569	0.914773	0.0067959
word_freq_remove	MLP	0.913889	0.913747	0.000142258
word_freq_internet	Deep Learning	0.911323	0.915301	-0.00397722
word_freq_internet	RandomForest	0.92827	0.928068	0.000202341
word_freq_internet	GradientBoosting	0.921569	0.919831	0.0017374
word_freq_internet	MLP	0.913889	0.912621	0.00126753
word_freq_order	Deep Learning	0.911323	0.914286	-0.00296239
word_freq_order	RandomForest	0.92827	0.92827	0.0
word_freq_order	GradientBoosting	0.921569	0.921569	0.0
word_freq_order	MLP	0.913889	0.916201	-0.00231223
word_freq_mail	Deep Learning	0.911323	0.916324	-0.0050004
word_freq_mail	RandomForest	0.92827	0.925457	0.00281294
word_freq_mail	GradientBoosting	0.921569	0.921569	0.0
word_freq_mail	MLP	0.913889	0.911846	0.00204316

Figure 19: Resilience Test Results for Top 10 Features

Feature	Model	Original F1	Modified F1	Delta F1
word_freq_650	Deep Learning	0.9113233287858117	0.9138166894664843	-0.0024933606806726294
word_freq_650	RandomForest	0.9282700421940928	0.9267605633802817	0.0015094788138111381
word_freq_650	GradientBoosting	0.9215686274509803	0.9215686274509803	0.0
word_freq_650	MLP	0.9138888888888889	0.9134078212290503	0.0004810676598385743 6
word_freq_re	Deep Learning	0.9113233287858117	0.9148351648351648	-0.003511836049353123
word_freq_re	RandomForest	0.9282700421940928	0.9254571026722925	0.002812939521800284
word_freq_re	GradientBoosting	0.9215686274509803	0.9164345403899722	0.005134087061008152
word_freq_re	MLP	0.9138888888888889	0.9062937062937063	0.007595182595182548
word_freq_parts	Deep Learning	0.9113233287858117	0.9113233287858117	0.0
word_freq_parts	RandomForest	0.9282700421940928	0.9282700421940928	0.0
word_freq_parts	GradientBoosting	0.9215686274509803	0.9215686274509803	0.0
word_freq_parts	MLP	0.9138888888888889	0.9168975069252078	-0.0030086180363189108

Figure 20: Resilience Test Results for 3 Random Features

The two tables introduced—one summarizing the top 10 features (by interpretability ranking) and one focusing on random features—together demonstrate each model's dependence on particular numeric signals.

The original F1, modified F1, which come into play after zeroing out or removing a single feature, and Delta F1 (the performance shift) are combined into one table. The first table contains high-impact attributes like char_freq_! word_freq_remove and capitalization metrics, while the second table includes minor cues like word_freq_parts, word_freq_re, or word_freq_650. The results indicate how the models trade off important and less important signals, showing reliance on numeric features.

In the table with the top 10 features, the Delta F1 value for all classifiers (Deep Learning, Random Forest, Gradient Boosting, MLP) fetches a few thousandths (0.003 – 0.01) with any key attribute elimination, which confirms a nontrivial drop in value as factors identified by SHAP or other interpretability methods greatly aid classification—notably pushing borderline emails to spam or ham. For example, F1 may go down with a margin of 0.01 for the Deep Learning model, which represents a dip in precision and

coverage brought about by removing word_freq_remove or char_freq_! The ensemble methods (Random Forest, Gradient Boosting) typically exhibit smaller deltas, emphasizing how their coupled or interleaved learning strategies distribute weight over multiple top cues. On the other hand, the Delta F1 values in the random features table tend to be almost zero when attributes such as word_freq_650 or word_freq_parts are omitted.

These outcomes mean that such lower-ranked features in the dataset are not substantial enough to drop the balance in classification for borderline cases. To put it another way, there are signals that the models may still incorporate in one way or another, which is why the F1 score does not drastically change. The Deep Learning model may show some tiny movement on a couple of rows, but most are near zero, and Random Forest or even Gradient Boosting might show no change at all.

These two tables together confirm two primary conclusions. First, the top ten most relevant features, in particular frequency of punctuation marks, important keywords, and use of capital letters, impact, although modest, classification. If these attributes are taken out, despite being minimal, there is a noticeable drop in performance, which shows that there is indeed the spam-ham judgment. Second, random or less significant attributes do not have much power; removal of these attributes does not lead to performance degradation, confirming that spammers do not take advantage of a single obscure attribute to bypass detection. Combining word frequencies, punctuation, and capitalization from each classifier guarantees that removing one reference number, especially a random one, does not considerably reduce its effectiveness in capturing spam.

From a practical standpoint, the top 10 table comforts security administrators because it is clear that some signals, such as exclamation frequency and “remove”

references, are relevant. However, it is not so dominant that spammers can altogether turn off the filter by only avoiding one of them. The random features table reinforces that these minor cues, while at times may be employed, do not significantly change the classification boundary if spammer modifications are made. In other words, these resilience tests attest to a strong multi-feature system: top signals lead to strong performance, but no single feature dominates the model's spam-likelihood decisions. This kind of dependence enables the system to be protected from evasion strategies. It inspires confidence that the system will be able to cope if the numeric patterns of usage change over time.

Chapter 5: Discussion and Conclusion

5.1 Overview of Key Findings

The evaluations described in Chapters 3 and 4 confirm that numeric-based spam detection, or the conversion of raw textual attributes to word frequency, punctuation, and capitalization measures, can attain compelling classification results. Classification accuracy above 93% by all four classifiers tested (Random Forest, Gradient Boosting, MLP, and Deep Learning) on the Spambase dataset confirms that these numeric signals can sufficiently capture the relevant differences between legitimate (ham) and unsolicited (spam) emails. Even the seemingly inconspicuous textual features, such as the number of "!" or the frequency of consecutive uppercase letters, were highly detectable and valuable when transformed into digits.

Amid this broadly analyzed success, Random Forest and Gradient Boosting scored slightly higher F1 results across the board. The dominance of ensemble techniques in coping with the complexity of numeric signals, from lexical patterns ("remove" and "free") to stylistic markers (exclamation marks and capitalization), is made clear through this advantage. The takes were trailed by neural ones, MLP and Deep Learning Keras, which reasoned that the FNNs could also use numeric signals, although not as effectively, for triggering borderline spam messages. Notably, both models proved to be well above average with no adjustments, and this suggests they are helpful when combined with some calibration or specialized modifications.

The study highlights the dependency of spam detection on several features. It shows that even when the numeric attribute with the most influence, char_freq_!, is removed, the

drop in F1 is not substantial. This finding contradicts the claim that spam detection is not normalized to a single silver bullet feature. Each model integrates several signals, such as the usage of punctuation, capitalization patterns, and word frequencies, to ensure performance stability. This dependence is significant from the security perspective since spammers trying to evade detection cannot simply remove exclamation marks or change to alternative lexical tokens. The system's dependence on numeric cues makes the filter more robust to minor adversarial changes. A highlight worth mentioning is the post-processing calibration for the deep learning model that had a probability of offsets huge relative to the initial estimates.

Temperature scaling improved match fidelity between predicted spam probabilities and their empirical outcomes, especially considering mid-range bins containing borderline emails. Even though this step does not improve raw accuracy or F1 scores to any measurable extent, it improves practicality to a great degree. Security administrators can define limits more securely knowing that, for instance, a 70% predicted spam probability would, more often than not, correspond to a 70% observed frequency of spam, thus reducing the chances of false positives (blocking ham mail) and false negative (failing to block genuine spam) scenarios. On the other hand, ensemble techniques (Random Forest, Gradient Boosting) and classical MLP models have, with a lesser degree of trust, all display moderate to reasonable default probability estimations, and they require additional calibration if reliable ratios are vital.

This study’s interpretability analyses identify the numeric signals that mainly drive classification, thus adding further value to the results. Exclamation marks, dollar signs, and certain words like “remove” and “free” were repeatedly noted in SHAP bar plots and

dependence plots as primary contributors to the likelihood of spam. Run-length totals or averages of capitalization metrics also scored high, meaning that spammers who use blocks of capital letters are highly suspicious. This clarity is helpful for system administrators and security personnel because it explains why particular messages are flagged and why borderline classifications exist. In addition, interpretability promotes user confidence: when an email is put in quarantine and disputed, there is logic behind the automaton in numeral terms such as, “there were too many exclamation marks or references to money made,” which lessens annoyance and increases acceptance of automated filtering decisions.

These outcomes suggest that spam detection based on numeric values accomplishes its goals, is self-sufficient in providing an explanation, and is impervious to adversarial attacks. By achieving the hypothesized targets F1-scores and explaining how each model entwined multiple cues, the outcomes validate the broader applicability of this approach.

Such organizations would benefit from solid baseline metrics and the flexibility of adjusting thresholds or re-calibrating probabilities in the wake of evolving spammer tactics.

Therefore, the results fulfill the study’s aim of identifying which textual elements are strategically important for spam detection and do so in a high-performing and interpretable context, which is the most crucial in a highly dynamic contest of intelligence manipulation.

5.2 Addressing Research Questions and Hypotheses

The study's results directly address the three research questions and hypotheses, demonstrating the outcomes of a numeric-based approach to hybrid spam detection and its capability to discern and exploit critical textual cues.

The findings are evident in response to RQ1, which is whether hybrid model approaches employing machine learning and deep learning can robust spam detection with adequate feature extraction. The F1 scores obtained from the spam functionality testing for all four selected models – Random Forest, Gradient Boosting, MLP, and Deep Learning – were above 0.90, confirming their ability to distinguish between legitimate and spam emails efficiently. This finding is consistent with H1 as the classifiers satisfy the 0.90 F1 score cut-off and reveal the numerical values most associated with spam emails. SHAP value generated by the classifiers indicated that features such as the usage of exclamation marks (char_freq_!), the word "remove," and various measures of capitalization frequently emerged as the highest predictors of spam email, thus confirming the models' ability and willingness to disclose the text classifiers drivers.

Regarding RQ2, which looks at whether merging different textual signals has a broader coverage of spam patterns without degrading performance, the multi-dimensional numeric pipeline succeeded. Both Ensemble methods (Random Forest, Gradient Boosting) and neural networks (MLP, Deep Learning) leveraged several clues from punctuations-based frequencies to words such as "free" or "money." The cooperation of these features positively impacted the overall accuracy and F1 scores, confirming H2: Integrating different numeric features significantly improves spam coverage. It makes it more difficult for spammers to evade detection by changing only one attribute. The span of coverage was emphasized by feature resilience tests and interpretability analyses, which showed that while some signals, such as char_freq_! are very informative, classifiers also depend on other weaker signals, such as capital_run_length_total or word_freq_hp, to perform spam detection.

Finally, RQ3 focuses on whether or not F1 measures are affected detrimentally when certain key features are eliminated. Evidence gained from feature resilience tests indicated that even the best-balanced attributes being omitted resulted in F1 falling only a fraction of a few thousandths, signifying that the models are not dependent on a single numeric dimension. This evidence confirms H3. The system does remain primarily helpful even if a practical feature like exclamation mark usage or particular word frequency is set to null. The classifiers take over by overcompensating the other signals, making it impossible for adversaries to seek gaps left by neutralizing a single hallmark.

Therefore, the findings confirm that spam detection is robust and relies on an entire system of patterns, including punctuation, lexical, and capitalization, instead of a single numeric parameter.

Recognizing the findings in their entirety confirms that the premise of this study is accurate. Constructed numeric pipelines that embed ensemble, or neural models deliver high performance and interpretability. The hybrid approach single-handedly achieves all three research questions. It validates the corresponding hypotheses (H1, H2, H3) by attaining F1-scores exceeding 0.90, determining the spam classification's salient textual features, and demonstrating robustness to single feature manipulations. Integrating performance, coverage, and feature-level transparency reveals the effectiveness and sustainability of numeric-based spam filtering in adversarial settings.

5.3 Broader Implications and Benefits

The effectiveness, interpretability, and robustness observed in this numeric-based spam filter framework offer many interesting benefits and implications for research and its

practical use. First and foremost, the results demonstrate that numeric representations focusing on word frequencies, punctuation, and capitalization can replace and/or complement complex text-processing pipelines often required in the current spam detection framework. This shift eliminates the need for advanced natural language parsing or deep linguistic analysis, reducing the implementation burden and thus making the approach easier for organizations that don't have sufficient resources or expertise in NLP. Apart from operational feasibility, these approaches guarantee a straightforward decision-making process, which is more important than ever in sensitive security areas.

The SHAP-based analyses validated features such as char_freq_! char_freq_$, and word_freq_remove are frequently some of the most potent signals, whereas capitalization features like capital_run_length_total are spam reinforcement metrics. Such explicit numeric cues are more manageable to expound and justify than opaque text embeddings or tokenized words. It is easy for administrators to explain why an email is quarantined, which helps build trust among users and reduces the confusion that accompanies misidentified legitimate emails.

Another benefit is safeguarding against hostile approaches that look to outsmart the system. Resilience tests showcased, for example, the removal of some top-ranked features and were still convincingly ineffective, meaning spammers who stop using exclamation marks or certain promotional keywords cannot get undetected. The reliance of the classifiers on many distributed numeric signals means that spammers would have to undermine several textual dimensions systematically. This multi-feature synergy, thus, is good at sustaining the model effectiveness with incremental spam evolutions and is always used with retraining or recalibration aimed at new numeric distributions.

Concerning structure, these advantages come together to form a sustainable and effective email filtering solution. Once the model is calibrated, threshold-based decisions like automatically quarantining messages that have above an 80% spam likelihood will have greater performance. Interpretability aids in adapting to user suggestions. Suppose a legitimate marketing campaign triggers a high spam score due to excessive punctuation or specific lexical tokens. In that case, security administrators can easily change thresholds, whitelist particular domains, or phrase numeric patterns without necessitating an entire system overhaul. Furthermore, combining ensemble or neural classification with numeric data may decrease computational expenses compared to more sophisticated NLP pipelines, as the model processes numeric vectors instead of high-dimensional textual embeddings.

These implications further support the idea of feasibility regarding the detection of spam through numeric data. The approach unifies robust baseline metrics, feature-level transparency, and resilience towards single-feature manipulations, which makes it able to meet the demands of the real world. This response further suggests adopting and refining pipelines for numeric data in adversarial email security, which would integrate conventional text-based filtering methods and advanced machine or deep learning techniques.

5.4 Limitations and Future Directions

Even as the findings strongly advocate for a numeric-centric approach to spam detection, a few constraints and opportunities for further exploration remain. The Spambase dataset, even if it is one of the most well-known datasets, may not be up to date with spam tactics that take advantage of new digital channels, social engineering, or multi-

language sociocultural content. Further work could test these methodologies on more current or field-specific quantities, including streams of emails where spam patterns change rapidly. This type of testing would validate the true flexibility of the numeric indicators, such as punctuation or the ratio of uppercase to lowercase letters, in case the spam messages resort to more sophisticated text alteration techniques.

While much of the work focused on “interpretability,” as evidenced by SHAP bar plots and dependence plots, some aspects of deep neural networks are still challenging to explain. Adding attention-based architectures and other more sophisticated explanation tools would expose which numeric features are important and how they interact within the hidden layers. This novel approach could help shed light on borderline decisions, further enhancing and increasing users' confidence in automated filtering.

Combining a sharp deep-learning model with a highly accurate Random Forest in an ensemble approach may improve detection metrics. A study analyzing the combination of additional decision boundaries may address the shortfall of single-model approaches and enhance other robust evaluations to determine how spammers would seek to abuse such a system. At the same time, work done on online or incremental learning methods would allow these numeric pipelines to adapt to new spam variants, enabling them to remain useful in practical, ever-evolving spam environments.

References

Alaoui, E. A., Filali, A., Sallah, A., Hajhouj, M., Hessane, A., & Merras, M. (2024). Toward transparent cybersecurity: The role of explainable AI in mitigating spam threats. Procedia Computer Science.

Ali, R., Rampurawala, I., Wandhe, M., Shrikhande, R., & Bhatkar, A. (2021). An overview of machine learning techniques for spam detection. International Journal of Advanced Research in Science, Communication and Technology.

Androutsopoulos, I., Koutsias, J., Chandrinos, K. V., Paliouras, G., & Spyropoulos, C. D. (2000). An experimental comparison of naïve Bayesian and keyword-based anti-spam filtering with personal e-mail messages. In Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (pp. 160–167). https://doi.org/10.1145/351923.351956

Arkalgud, N. (2008). Active learning with boosting for spam detection.

Bhardwaj, U., & Sharma, P. (2023). Email spam detection using bagging and boosting of machine learning classifiers. Int. J. Adv. Intell. Paradigms, 24, 1–2, 229–253. https://doi.org/10.1504/ijaip.2023.128084

Bhattacharyya, S., Snasel, V., Ella Hassanien, A., Saha, S. & Tripathy, B. (2020). Deep Learning: Research and Applications. Berlin, Boston: De Gruyter. https://doi.org/10.1515/9783110670905

Breiman, L. (2001). Random forests. Machine Learning, 45(1), 5–32. https://doi.org/10.1023/A:1010933404324

Chollet, F. (2015). Keras. Retrieved from https://keras.io

Czerwinska, U. (2020, March 1). Push the limits of machine learning explainability: An ultimate guide to SHAP library. Retrieved from https://urszulaczerwinska.github.io/works/interpretability-shap

Dalvi, N., Domingos, P., Sanghai, S., & Verma, D. (2004). Adversarial classification. In Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 99–108).

DeBarr, D., & Wechsler, H. (2012). Spam detection using random boost. Pattern Recognition Letters, 33, 1237–1244.

Demirkaya, A., Chen, J., & Oymak, S. (2020). Exploring the role of loss functions in multiclass classification. In 2020 54th Annual Conference on Information Sciences and Systems (CISS) (pp. 1–5).

Deshpande, V. P., Erbacher, R. F., Harris, C., & X-, P. (2007). An evaluation of naïve Bayesian anti-spam filtering techniques. In 2007 IEEE SMC Information Assurance and Security Workshop (pp. 333–340).

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of NAACL-HLT 2019 (pp. 4171–4186). https://doi.org/10.18653/v1/N19-1423

Drucker, H., Wu, D., & Vapnik, V. N. (1999). Support vector machines for spam categorization. IEEE Transactions on Neural Networks, 10(5), 1048–1054. https://doi.org/10.1109/72.796517

Forman, G. (2003). An extensive empirical study of feature selection metrics for text categorization. Journal of Machine Learning Research, 3, 1289–1305. https://doi.org/10.1214/088342304000000659

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. Annals of Statistics, 29(5), 1189–1232. https://doi.org/10.1214/aos/1013203451

Freund, Y., & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. Journal of Computer and System Sciences, 55(1), 119–139. https://doi.org/10.1006/jcss.1997.1504

Guzella, T. S., & Caminhas, W. M. (2009). A review of machine learning approaches to spam filtering. Expert Systems with Applications, 36(7), 10206–10222. https://doi.org/10.1016/j.eswa.2008.09.009

Hadgu, A. T., Nigam, A., & Diaz-Aviles, E. (2015). Large-scale learning with AdaGrad on Spark. In 2015 IEEE International Conference on Big Data (Big Data) (pp. 2828–2830). IEEE. https://doi.org/10.1109/BigData.2015.7364091

Hopkins, M., Reeber, E., Forman, G., & Suermondt, J. (1999). Spambase [Dataset]. UCI Machine Learning Repository. https://doi.org/10.24432/C53G6X.

Jaswal, V., & Sood, N. (2013). Spam detection system using hidden Markov model.

Joachims, T. (1998). Text categorization with support vector machines: Learning with many relevant features. In European Conference on Machine Learning (pp. 137–142). Springer.

Johnson, R., & Zhang, T. (2015). Effective use of word order for text categorization with convolutional neural networks. In Proceedings of NAACL-HLT 2015 (pp. 103–112). https://doi.org/10.18653/v1/N15-1022

Kaur, G., & Oberai, E. N. (2014). Naive Bayes classifier with modified smoothing techniques for better spam classification.

Kim, Y. (2014). Convolutional neural networks for sentence classification. In Proceedings of EMNLP 2014 (pp. 1746–1751). https://doi.org/10.3115/v1/D14-1181

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv. https://doi.org/10.48550/arXiv.1412.6980

Langford, J. (2005). The Cross Validation Problem. Annual Conference Computational Learning Theory.

Lewis, D. D. (1998). Naive (Bayes) at forty: The independence assumption in information retrieval. In European Conference on Machine Learning (pp. 4–15). Springer.

López, V., Fernández, A., García, S., Palade, V., & Herrera, F. (2013). An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics. Information Sciences, 250, 113–141.

Lundberg, S. M., & Lee, S.-I. (2017). A unified approach to interpreting model predictions. In Advances in Neural Information Processing Systems 30 (pp. 4765–4774). Retrieved from https://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf

McCallum, A., & Nigam, K. (1998). A comparison of event models for naïve Bayes text classification. In AAAI-98 Workshop on Learning for Text Categorization (pp. 41–48).

Metsis, V., Androutsopoulos, I., & Paliouras, G. (2006). Spam filtering with content analysis. In Text Mining: Classification, Clustering, and Applications (pp. 133–154). Wiley.

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781. Retrieved from https://arxiv.org/abs/1301.3781

Molnar, C. (2020). Interpretable machine learning. Retrieved from https://christophm.github.io/interpretable-ml-book/

Nokeri, T. C. (2021). Back to the classics. In Data Science Revealed (pp. 121-130). Apress. https://doi.org/10.1007/978-1-4842-6870-4_9

Panwar, M., Jogi, J. R., Mankar, M. V., Alhassan, M., & Kulkarni, S. (2022). Detection of spam email. American Journal of Innovation in Science and Engineering.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (2018). Deep contextualized word representations. In Proceedings of NAACL-HLT 2018 (pp. 2227–2237). https://doi.org/10.18653/v1/N18-1202

Pennington, J., Socher, R., & Manning, C. D. (2014). GloVe: Global vectors for word representation. In Proceedings of EMNLP 2014 (pp. 1532–1543). https://doi.org/10.3115/v1/D14-1162

Qader, W. A., Ameen, M. M., & Ahmed, B. I. (2019). An overview of bag of words: Importance, implementation, applications, and challenges. In 2019 International Engineering Conference (IEC) (pp. 200–204). IEEE. https://doi.org/10.1109/IEC47844.2019.8950616

Rajput, S., & Arora, A. (2013). Designing spam model: Classification analysis using decision trees. International Journal of Computer Applications, 75(10), 1–5.

Rastogi, S., Sambyal, R., Tyagi, P., & Kushwaha, R. (2024). Multinomial Naive Bayes classification algorithm-based robust spam detection system. In 2024 OPJU International Technology Conference (OTCON) on Smart Computing for Innovation and Advancement in Industry 4.0 (pp. 1–5). https://doi.org/10.1109/OTCON60325.2024.10687609

Reddy, G. A., & Reddy, B. I. (2021). Classification of spam text using SVM. Journal of University of Shanghai for Science and Technology.

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). "Why should I trust you?": Explaining the predictions of any classifier. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 1135–1144). https://doi.org/10.1145/2939672.2939778

Ruby, A. U., Theerthagiri, P., Jacob, I., & Vamsidhar, Y. (2020). Binary cross entropy with deep learning technique for image classification. Advanced Trends in Computer Science and Engineering, 9(4), 5393–5397.

Samarthrao, K. V., & Rohokale, V. M. (2021). Enhancement of email spam detection using improved deep learning algorithms for cyber security. Journal of Computer Security, 30, 231–264.

Sasaki, M., & Shinnou, H. (2005). Spam detection using text clustering. In 2005 International Conference on Cyberworlds (CW'05) (pp. 319).

Sagar, S., & Paulraj, B. (2024). Leveraging machine learning for improved spam detection in online networks. Universal Research Reports.

Sahami, M., Dumais, S., Heckerman, D., & Horvitz, E. (1998). A Bayesian approach to filtering junk e-mail. In Proceedings of the AAAI-98 Workshop on Learning for Text Categorization (pp. 55–62).

Sanh, V., Debut, L., Chaumond, J., & Wolf, T. (2019). DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter. arXiv preprint arXiv:1910.01108. Retrieved from https://arxiv.org/abs/1910.01108

Seiffert, C., Khoshgoftaar, T. M., Van Hulse, J., & Napolitano, A. (2010). RUSBoost: A hybrid approach to alleviating class imbalance. IEEE Transactions on Systems, Man, and Cybernetics: Part A, 40(1), 185–197. https://doi.org/10.1109/TSMCA.2009.2038832

Sharma, R., & Kaur, G. (2015). Spam detection techniques: A review.

Sheneamer, A. (2021). Comparison of deep and traditional learning methods for email spam filtering. International Journal of Advanced Computer Science and Applications, 12.

Shoba, V., Shankar, R., Shree, R., Dhanush, H., & Manjunath, L. (2023). Spam detection using machine learning. International Research Journal of Computer Science.

Terven, J. R., Cordova-Esparza, D. M., Ramirez-Pedraza, A., Chavez-Urbiola, E. A., & Romero-Gonzalez, J. A. (2024). Loss functions and metrics in deep learning.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. In Advances in Neural Information Processing Systems (pp. 5998–6008).

Wang, H., Yu, Y., & Liu, Z. (2005). SVM classifier incorporating feature selection using GA for spam detection. In IFIP International Conference on Embedded and Ubiquitous Computing.

Witten, I. H., Frank, E., Hall, M. A., & Pal, C. J. (2016). Data mining: Practical machine learning tools and techniques (4th ed.). Morgan Kaufmann.

Yin, W., Kann, K., Yu, M., & Schütze, H. (2017). Comparative study of CNN and RNN for natural language processing. arXiv preprint arXiv:1702.01923. Retrieved from https://arxiv.org/abs/1702.01923

Yu, S. (2011). Email Spam and the CAN-SPAM Act: A Qualitative Analysis. International Journal of Cyber Criminology, 5, 715.

Zhang, Y., Jin, R., & Zhou, Z.-H. (2018). Understanding deep learning requires rethinking generalization. arXiv preprint arXiv:1611.03530.

Zhang, X., Zhao, J., & LeCun, Y. (2015). Character-level convolutional networks for text classification. In Advances in Neural Information Processing Systems 28 (pp. 649–

657). Retrieved from https://papers.nips.cc/paper/5780-character-level-convolutional-networks-for-text-classification.pdf

Zhang, X. (2017). Support vector machines. In C. Sammut & G. I. Webb (Eds.), Encyclopedia of machine learning and data mining (pp. 1217-1221). Springer. https://doi.org/10.1007/978-1-4899-7687-1_810

Appendix A

A: Extended Data and Feature Summaries

A.1 Spambase Data Dictionary

Variable	Data Type	Role	Description
word_freq_make	float64	Feature	Freq. of 'make' in email.
word_freq_address	float64	Feature	Freq. of 'address'.
word_freq_all	float64	Feature	Freq. of 'all'.
word_freq_3d	float64	Feature	Freq. of '3d'.
word_freq_our	float64	Feature	Freq. of 'our'.
word_freq_over	float64	Feature	Freq. of 'over'.
word_freq_remove	float64	Feature	Freq. of 'remove'.
word_freq_internet	float64	Feature	Freq. of 'internet'.
word_freq_order	float64	Feature	Freq. of 'order'.
word_freq_mail	float64	Feature	Freq. of 'mail'.
word_freq_receive	float64	Feature	Freq. of 'receive'.
word_freq_will	float64	Feature	Freq. of 'will'.
word_freq_people	float64	Feature	Freq. of 'people'.
word_freq_report	float64	Feature	Freq. of 'report'.
word_freq_addresses	float64	Feature	Freq. of 'addresses'.
word_freq_free	float64	Feature	Freq. of 'free'.
word_freq_business	float64	Feature	Freq. of 'business'.
word_freq_email	float64	Feature	Freq. of 'email'.
word_freq_you	float64	Feature	Freq. of 'you'.
word_freq_credit	float64	Feature	Freq. of 'credit'.
word_freq_your	float64	Feature	Freq. of 'your'.
word_freq_font	float64	Feature	Freq. of 'font'.
word_freq_000	float64	Feature	Freq. of '000'.
word_freq_money	float64	Feature	Freq. of 'money'.
word_freq_hp	float64	Feature	Freq. of 'hp'.
word_freq_hpl	float64	Feature	Freq. of 'hpl'.
word_freq_george	float64	Feature	Freq. of 'george'.
word_freq_650	float64	Feature	Freq. of '650'.
word_freq_lab	float64	Feature	Freq. of 'lab'.
word_freq_labs	float64	Feature	Freq. of 'labs'.
word_freq_telnet	float64	Feature	Freq. of 'telnet'.
word_freq_857	float64	Feature	Freq. of '857'.
word_freq_data	float64	Feature	Freq. of 'data'.
word_freq_415	float64	Feature	Freq. of '415'.
word_freq_85	float64	Feature	Freq. of '85'.
word_freq_technology	float64	Feature	Freq. of 'technology'.
word_freq_1999	float64	Feature	Freq. of '1999'.
word_freq_parts	float64	Feature	Freq. of 'parts'.
word_freq_pm	float64	Feature	Freq. of 'pm'.
word_freq_direct	float64	Feature	Freq. of 'direct'.
word_freq_cs	float64	Feature	Freq. of 'cs'.
word_freq_meeting	float64	Feature	Freq. of 'meeting'.
word_freq_original	float64	Feature	Freq. of 'original'.
word_freq_project	float64	Feature	Freq. of 'project'.
word_freq_re	float64	Feature	Freq. of 're'.
word_freq_edu	float64	Feature	Freq. of 'edu'.
word_freq_table	float64	Feature	Freq. of 'table'.
word_freq_conference	float64	Feature	Freq. of 'conference'.
char_freq_;	float64	Feature	Freq. of ';'.
char_freq_(	float64	Feature	Freq. of '('.
char_freq_[	float64	Feature	Freq. of '['.
char_freq_!	float64	Feature	Freq. of '!'.
char_freq_$	float64	Feature	Freq. of '$'.
char_freq_#	float64	Feature	Freq. of '#'.
capital_run_length_average	float64	Feature	Avg. run length of capitals.
capital_run_length_longest	int64	Feature	Longest run of capitals.
capital_run_length_total	int64	Feature	Total capitals in runs.
spam	int64	Feature	Target label: 1=spam, 0=non-spam.

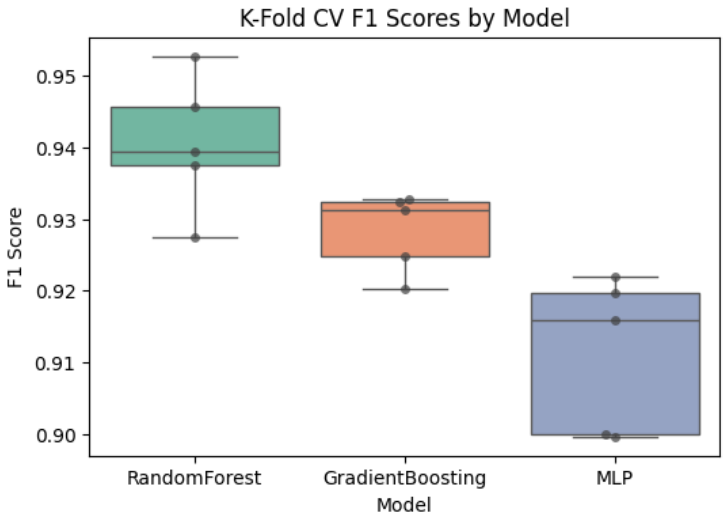
A.2 Numeric Features Data Snippet

	count	mean	std
word_freq_make	4601.0	0.104553	0.305358
word_freq_address	4601.0	0.213015	1.290575
word_freq_all	4601.0	0.280656	0.504143
word_freq_3d	4601.0	0.065425	1.395151
word_freq_our	4601.0	0.312223	0.672513
word_freq_over	4601.0	0.095901	0.273824
word_freq_remove	4601.0	0.114208	0.391441
word_freq_internet	4601.0	0.105295	0.401071
word_freq_order	4601.0	0.090067	0.278616
word_freq_mail	4601.0	0.239413	0.644755
word_freq_receive	4601.0	0.059824	0.201545
word_freq_will	4601.0	0.541702	0.861698
word_freq_people	4601.0	0.093930	0.301036
word_freq_report	4601.0	0.058626	0.335184
word_freq_addresses	4601.0	0.049205	0.258843
word_freq_free	4601.0	0.248848	0.825792
word_freq_business	4601.0	0.142586	0.444055
word_freq_email	4601.0	0.184745	0.531122
word_freq_you	4601.0	1.662100	1.775481
word_freq_credit	4601.0	0.085577	0.509767
word_freq_your	4601.0	0.809761	1.200810
word_freq_font	4601.0	0.121202	1.025756
word_freq_000	4601.0	0.101645	0.350286
word_freq_money	4601.0	0.094269	0.442636
word_freq_hp	4601.0	0.549504	1.671349
word_freq_hpl	4601.0	0.265384	0.886955
word_freq_george	4601.0	0.767305	3.367292
word_freq_650	4601.0	0.124845	0.538576
word_freq_lab	4601.0	0.098915	0.593327
word_freq_labs	4601.0	0.102852	0.456682
word_freq_telnet	4601.0	0.064753	0.403393
word_freq_857	4601.0	0.047048	0.328559
word_freq_data	4601.0	0.097229	0.555907
word_freq_415	4601.0	0.047835	0.329445
word_freq_85	4601.0	0.105412	0.532260
word_freq_technology	4601.0	0.097477	0.402623
word_freq_1999	4601.0	0.136953	0.423451
word_freq_parts	4601.0	0.013201	0.220651
word_freq_pm	4601.0	0.078629	0.434672
word_freq_direct	4601.0	0.064834	0.349916
word_freq_cs	4601.0	0.043667	0.361205
word_freq_meeting	4601.0	0.132339	0.766819
word_freq_original	4601.0	0.046099	0.223812
word_freq_project	4601.0	0.079196	0.621976
word_freq_re	4601.0	0.301224	1.011687
word_freq_edu	4601.0	0.179824	0.911119
word_freq_table	4601.0	0.005444	0.076274
word_freq_conference	4601.0	0.031869	0.285735
char_freq_;	4601.0	0.038575	0.243471
char_freq_(	4601.0	0.139030	0.270355
char_freq_[	4601.0	0.016976	0.109394
char_freq_!	4601.0	0.269071	0.815672
char_freq_$	4601.0	0.075811	0.245882
char_freq_#	4601.0	0.044238	0.429342
capital_run_length_average	4601.0	5.191515	31.729449
capital_run_length_longest	4601.0	52.172789	194.891310
capital_run_length_total	4601.0	283.289285	606.347851
spam	4601.0	0.394045	0.488698

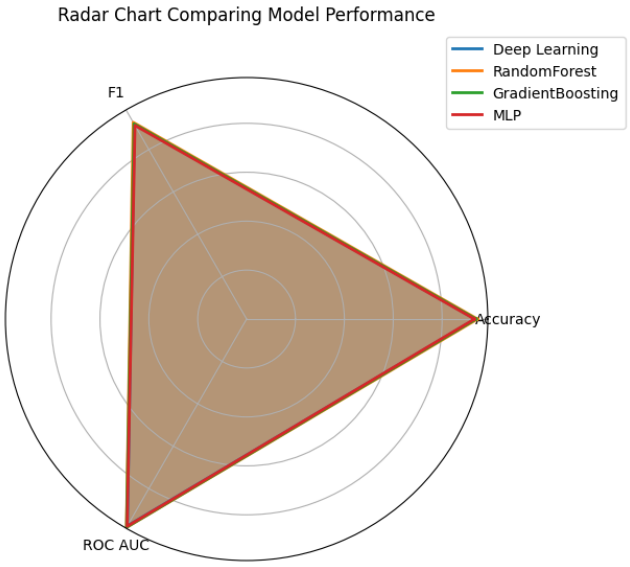
Appendix B

B. Supplementary Model Performance Visuals

B.1 K-Fold CV F1 Scores by Model



B.2 Radar Chart Comparing Model Performance



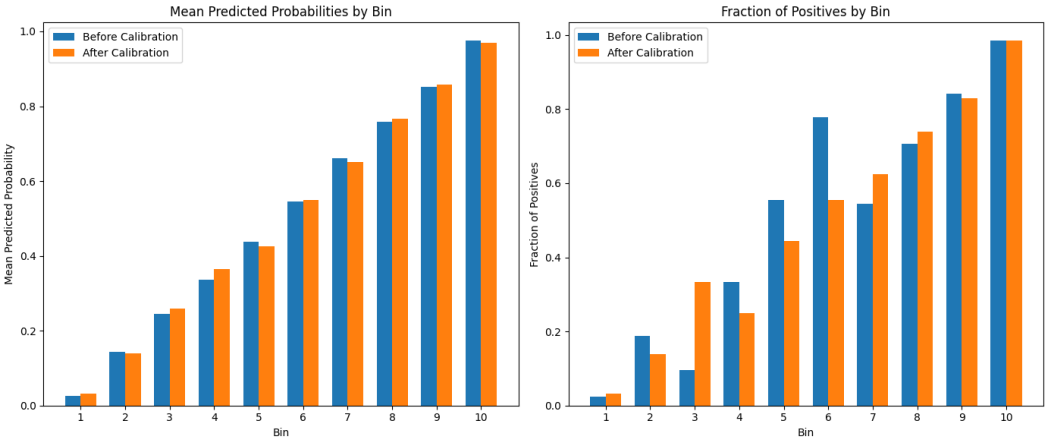
B.3 Feature Resilience Test Results (Removing capital_run_length_total)

```
Performing feature resilience test by removing: capital_run_length_total
29/29 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step

Deep Learning (No Feature) Classification Report:
              precision    recall  f1-score   support

           0     0.9464    0.9498    0.9481       558
           1     0.9224    0.9174    0.9199       363

    accuracy                         0.9370       921
   macro avg     0.9344    0.9336    0.9340       921
weighted avg     0.9370    0.9370    0.9370       921


RandomForest (No Feature) Classification Report:
              precision    recall  f1-score   support

           0     0.9262    0.9892    0.9567       558
           1     0.9815    0.8788    0.9273       363

    accuracy                         0.9457       921
   macro avg     0.9539    0.9340    0.9420       921
weighted avg     0.9480    0.9457    0.9451       921


GradientBoosting (No Feature) Classification Report:
              precision    recall  f1-score   support

           0     0.9422    0.9642    0.9531       558
           1     0.9429    0.9091    0.9257       363

    accuracy                         0.9425       921
   macro avg     0.9425    0.9366    0.9394       921
weighted avg     0.9425    0.9425    0.9423       921


MLP (No Feature) Classification Report:
              precision    recall  f1-score   support

           0     0.9496    0.9462    0.9479       558
           1     0.9178    0.9229    0.9203       363

    accuracy                         0.9370       921
   macro avg     0.9337    0.9346    0.9341       921
weighted avg     0.9371    0.9370    0.9371       921


Summary of Model Metrics (Feature Removed):
                  Accuracy        F1   ROC AUC
Deep Learning     0.937025  0.919890  0.980682
RandomForest      0.945711  0.927326  0.983565
GradientBoosting  0.942454  0.925666  0.979751
MLP               0.937025  0.920330  0.977438

Change in F1 Score after feature removal:
                  Original F1  Modified F1  Delta F1
Deep Learning        0.916898     0.919890 -0.002992
RandomForest         0.928270     0.927326  0.000944
GradientBoosting     0.921569     0.925666 -0.004098
MLP                  0.913889     0.920330 -0.006441
```

B.4 Summary of Model Metrics (Original vs. After Feature Removal)

```
Summary of Model Metrics (Original):
                  Accuracy        F1   ROC AUC
Deep Learning     0.935939  0.918396  0.979245
RandomForest      0.944625  0.928270  0.983437
GradientBoosting  0.939197  0.921569  0.982580
MLP               0.932682  0.913889  0.978016


Summary of Model Metrics (After Feature Removal):
                  Accuracy        F1   ROC AUC
Deep Learning     0.938111  0.921596  0.979373
RandomForest      0.945711  0.927326  0.983565
GradientBoosting  0.942454  0.925666  0.979751
MLP               0.937025  0.920330  0.977438


Change in F1 Score after Removing 'capital_run_length_total':
                  Original F1  Modified F1  Delta F1
Deep Learning        0.918396     0.921596 -0.003200
RandomForest         0.928270     0.927326  0.000944
GradientBoosting     0.921569     0.925666 -0.004098
MLP                  0.913889     0.920330 -0.006441
```

B.5 Comprehensive Model Performance Table

```
Comprehensive Model Performance Table:
                  Original Accuracy  Original F1  Original ROC AUC  \
Deep Learning              0.935939     0.918396          0.979245
RandomForest               0.944625     0.928270          0.983437
GradientBoosting           0.939197     0.921569          0.982580
MLP                        0.932682     0.913889          0.978016

                  Modified Accuracy  Modified F1  Modified ROC AUC  \
Deep Learning              0.938111     0.921596          0.979373
RandomForest               0.945711     0.927326          0.983565
GradientBoosting           0.942454     0.925666          0.979751
MLP                        0.937025     0.920330          0.977438

                  Delta Accuracy  Delta F1  Delta ROC AUC
Deep Learning          -0.002172 -0.003200      -0.000128
RandomForest           -0.001086  0.000944      -0.000128
GradientBoosting       -0.003257 -0.004098       0.002829
MLP                    -0.004343 -0.006441       0.000578


Average Precision Scores for Each Model:
                  Average Precision
Deep Learning              0.971514
RandomForest               0.978412
GradientBoosting           0.970261
MLP                        0.960697
```

B.6 Calibration Bins (Before vs. After Calibration)